

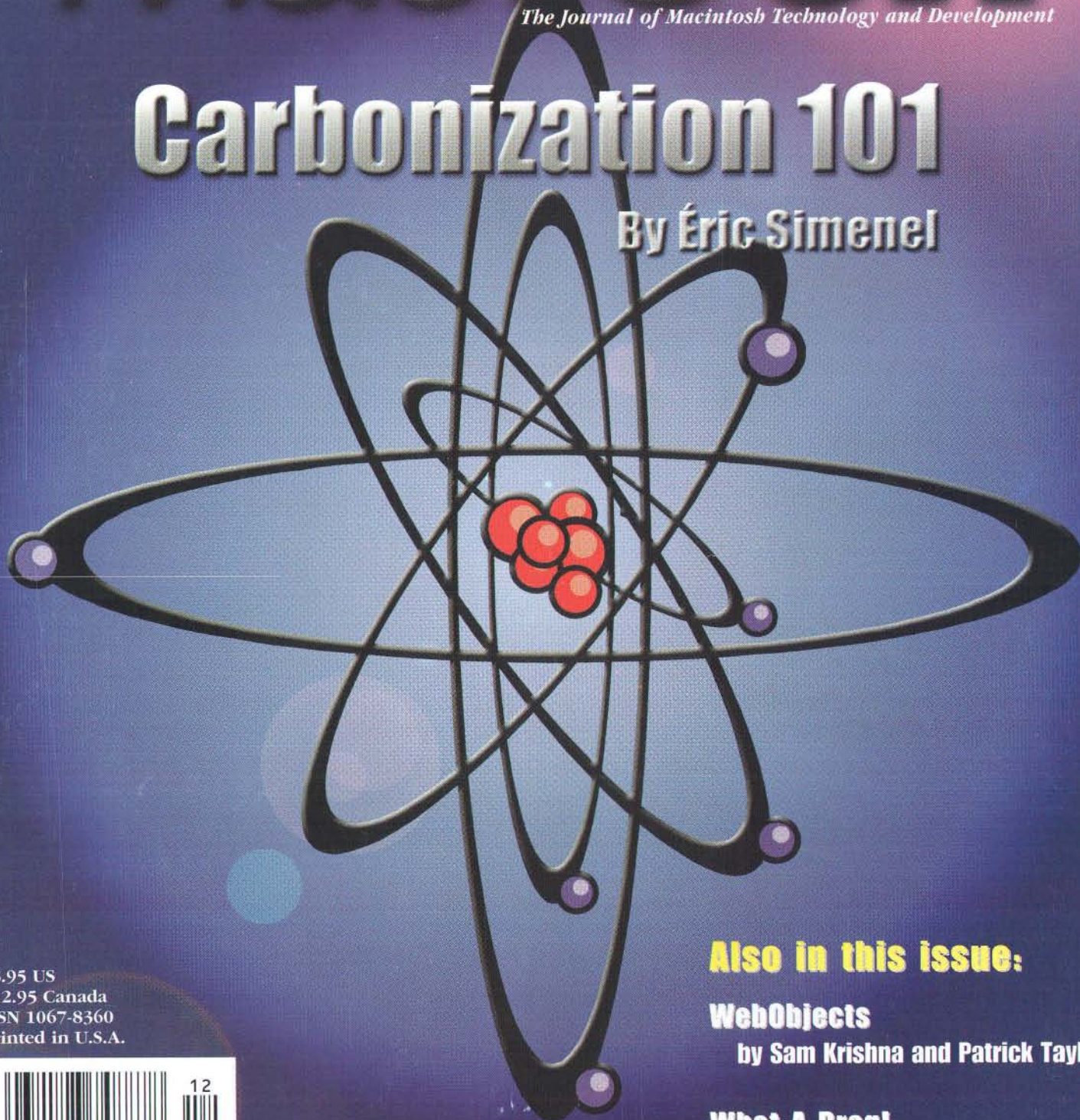
MacTech Magazine
Vol. 16, No. 12 • December 2000

MacTech[®]

The Journal of Macintosh Technology and Development

Carbonization 101

By Éric Simenel



\$8.95 US
\$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.



Also in this issue:

WebObjects

by Sam Krishna and Patrick Taylor

What A Drag!

by Andrew C. Stone

*"Never forget that the most powerful
force on earth is love."*

—Nelson A. Rockefeller

Move over love.



Get a new Apple PowerMac™ G4
or one of our refurbished models
at price you will L-O-V-E.



**Small Dog
Electronics**

1673 Main Street
Waitsfield, VT 05673 USA
Phone: **802-496-7171**
Online: **smalldog.com**



Apple Specialist

***"Without a doubt, the Premiere Resource Editor
for the Mac OS ... A wealth of time-saving tools."***

– MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."

– MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."

– Leonard Rosenthol, Aladdin Systems

***"Without Resorcerer, our localization efforts would look like a
Tower of Babel. Don't do product without it!"***

– Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."

– Bill Goodman, author of Smaller Installer and Compact Pro

"Resorcerer Rocks! Buy it, you will NOT regret it."

– Joe Zobkiw, author of A Fragment of Your Imagination

"Resorcerer will pay for itself many times over in saved time and effort."

– MacUser review

"The template that disassembles PICT's is awesome!"

– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensable in its own creation!"

– Doug McKenna, author of Resorcerer



Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater,
1.5MB RAM, CD-ROM

Standard price: \$256 (decimal)

Website price: \$128 - \$256

(Educational, quantity, or
other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMAESTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New
in
2.0:

- Very fast, HFS browser for viewing file tree of all volumes
- Extensibility for new Resorcerer Apprentices (CFM plug-ins)
- New AppleScript Dictionary ('aete') Apprentice Editor
- MacOS 8 Appearance Manager-savvy Control Editor
- PowerPlant text traits and menu command support
- Complete AIFF sound file disassembly template
- Big-, little-, and even mixed-endian template parsing
- Auto-backup during file saves; folder attribute editing
- Ships with PowerPC native, fat, and 68K versions

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**?

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS

Orders, Circulation, & Customer Service

Press Releases

Ad Sales

Editorial

Programmer's Challenge

Online Support

Accounting

Marketing

General

Web Site (articles, info, URLs and more...)

E-Mail/URL

cust_service@mactech.com

press_releases@mactech.com

ad_sales@mactech.com

editorial@mactech.com

prog_challenge@mactech.com

online@mactech.com

accounting@mactech.com

marketing@mactech.com

info@mactech.com

<http://www.mactech.com>

The MacTech Editorial Staff

Publisher • Neil Ticktin

Managing Editor • Jessica Stubblefield

Online Editor • Jeff Clites

Regular Columnists

Getting Started – Networking

by John C. Welch

Programmer's Challenge

by Bob Boonstra

MacTech Online

by Jeff Clites

From the Factory Floor

by Metrowerks

Regular Contributors

Vicki Brown, Andrew Stone,
Tim Monroe, Erick Tejkowski,
Kas Thomas, Will Porter,
Paul E. Seving, Tom Djajadiningrat,
and Jordan Dea-Mattson

MacTech's Board of Advisors

Dave Mark, Jordan Dea-Mattson,
Jim Straus, Jon Wiederspan,
and Eric Gundrum

MacTech's Contributing Editors

- Jim Black
- Michael Brian Bentley
- Tantek Çelik, Microsoft Corporation
- Marshall Clow
- John C. Daub
- Tom Djajadiningrat
- Bill Doerrfeld, Blueworld
- Andrew S. Downs
- Richard V. Ford, Packeteer
- Gordon Garb, Cobalt Networks
- John Hanay, Apple Computer
- Lorca Hanns, San Francisco State University
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Scott Knaster, Microsoft
- Mark Kriegsman, Clearway Technologies
- Peter N. Lewis, Stairways Software
- Bill McGlasson, Apple Computer
- Rich Morin
- Terje Norderhaug, Media*Design-In-Progress
- Nathan Nunn, Purity Software
- John O'Fallon, Maxum Development
- Alan Oppenheimer, Open Door Networks
- Avi Rappoport, Search Tools Consulting
- Ty Shipman, Kagi
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Dori Smith
- Andrew C. Stone, www.stone.com
- Michael Swan, Neon Software
- Chuck Von Rospach, Plaidworks
- Bill von Hagen
- Eric Zelenka

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • W2 Graphics

Marketing Managers

Nick DeMello and Alyse Yourist

Events Manager • Susan M. Worley

Network Administrator • Ben Baumer

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane

Susan Pomrantz

Shipping/Receiving • Joel Licardie

Board of Advisors • Steven Geller, Blake Park,
and Alan Carsrud

All contents are Copyright 1984-2000 by Xplain Corporation. All rights reserved. MacTech, Developer Depot, and Sprocket are registered trademarks of Xplain Corporation. Depot, The Depot, Depot Store, Video Depot, MacDev-I, THINK Reference, NetProfessional, NetProLive, JavaTech, WebTech, BeTech, ExplainIt, and the MacTutorMan are trademarks of Xplain Corporation. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders. Xplain Corporation does not assume any liability for errors or omissions by any of the advertisers, in advertising content, editorial, or other content found herein. Opinions or expressions stated herein are not necessarily those of the publisher and therefore, publisher assumes no liability in regards to said statements.



This publication is
printed on paper with
recycled content.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

C o n t e n t s

December 2000 • Volume 16, Issue 12

Feature Articles

WEBOBJECTS

- 22 EOKeyValueCoding**
by Sam Krishna and Patrick Taylor

QUICKTIME TOOLKIT

- 49 Timecode**
by Tim Monroe

MAC OS X

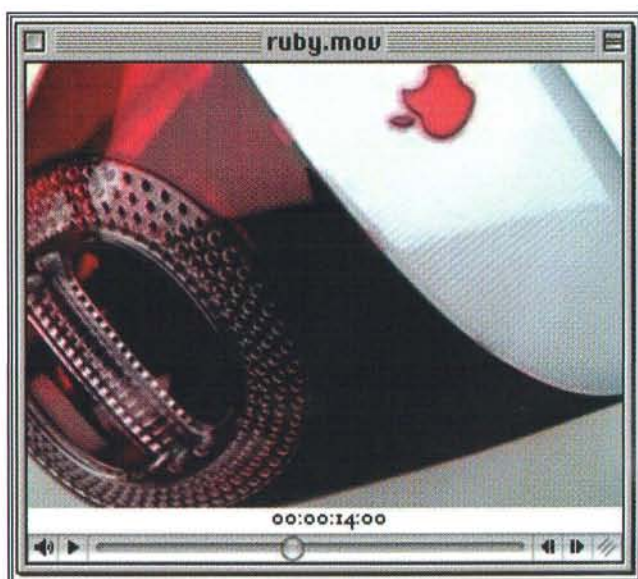
- 60 What A Drag!**
by Andrew C. Stone

PROGRAMMER'S BOOKSHELF

- 4 The C++ Programming Language**
by Paul E. Seving

QUICKDRAW 3D TRICKS

- 27 Cubby: Multiscreen Desktop VR, Part III**
*by Maarten Gribnau
and Tom Djajadiningrat*



QuickTime Toolkit Page 49

Carbon Compatibility Report

Report for: TveServicesForUnicode
File Creator: ????
File Type: APPL

Report date: 8/1/2000
Data gathered by analyzer version: 1.3
Data gathered on: 8/1/2000

Summary

Test	Number	Imports from Universal Interfaces
Analysis of Imports		
Supported API	152	Supported - 85.4%
Reserved with Modifications	1	Modified - 0.6%
Reserved But Not Recommended	7	Not Recommended - 3.9%
Unsupported API	18	Unsupported - 10.1%
Analysis of Access to Low Memory Addresses		
Analysis of Resources Loaded into System Heap		

Carbonization 101 Page 10

C o l u m n s

GETTING STARTED NETWORKING

- 6 MacNFS**
*by John C. Welch
Edited by Ilene Hoffman*

- 39 PROGRAMMER'S CHALLENGE**
Crutches
by Bob Boonstra

- 84 MACTECH ONLINE**
The New Basics
by Jeff Clites

COVER STORY

- 10 Carbonization 101**
by Éric Simenel

By Paul E. Sevinç, Switzerland

The C++ Programming Language

Third Edition and Special Edition

A couple of months ago, Addison-Wesley published a special edition of the third edition of *Bjarne Stroustrup's The C++ Programming Language* [1, 2]. The special edition features two new appendices that my copy of the third edition (9th printing) does not contain. However, both appendices are freely available from Bjarne Stroustrup's home page [3]. And except for minor differences that exist between any two printings (cf. erratas [3]), the contents of both editions are identical.

The book teaches Standard C++ and the C++ Standard Library, and discusses programming techniques supported by C++. In my experience, it also serves well as a reference manual.

The four main parts of the book—*Basic Facilities*, *Abstraction Mechanisms*, *The Standard Library*, and *Design Using C++* — are preceded by an introductory part, *Introductory Material*. All the chapters contain a section called *Advice* where in other books one would find a summary. With the exception of the first two chapters of the last part, all the chapters in the main parts end with exercise sections.

The advice sections just contain a list of tips (a few up to two dozen). Examples are “*Use 0 rather than NULL*” [1, p. 104] or “*Avoid type fields*” [1, p. 324]. Taken out of context, they appear to be trivial at best, but I find them to be quite valuable both while learning C++ and in practice.

INTRODUCTORY MATERIAL

The introduction consists of three chapters.

The first chapter is mostly non-technical and gives an overview of the book, recaps C++'s history, etc. Even though it gives the impression of being unimportant, I highly recommend to carefully read this chapter. It contains very interesting background material on C++.

The second and third chapter give an overview of the language and the standard library, respectively. What is quickly explained in these two chapters is explained in (much) more detail in later chapters. Readers new to C++ will most certainly not understand everything, but that is not necessary. However, who thinks of herself or himself as a C++ programmer can use these chapters as a refresher course on the one hand and as a fitness test on the other hand.

BASIC FACILITIES

The six chapters of part I describe the language constructs that support procedural and modular programming. These two paradigms are briefly explained in chapter 2 only. Therefore, the reader should at least be familiar with procedural programming and the principle of information hiding.

Chapter 4 presents the built-in types as well as enumerations, declarations, and synonyms for types (typedefs). Chapter 5 treats pointers and references, arrays, and structures (records). Chapter 6 discusses expressions,

Continued on page 86

Paul E. Sevinç became the 43rd president of the United States after Florida wasn't able to make up its mind. You can reach him at paul.sevinc@ubilab.org.

The key to thinking different...



**Works
on the
iMac!**

The New MacHASP USB Key!

Question: Is MacHASP USB* a software security key or a sales tool?

Answer: It's both!

MacHASP USB is the world's first software protection key for the iMac. It gives you sophisticated license enforcement and comprehensive protection against illegal use ... in other words, real security.

Then it gives you a big selling advantage.

With MacHASP USB, you can license multiple software modules and applications. You can instantly unlock or upgrade them in the field. Plus you can freely distribute demos.

Bottom line: MacHASP USB locks out illegal users and unlocks your

full sales potential – without getting in anyone's way. Call now to request a Developer's Kit and our newly published guide to USB features and benefits.

*For all USB-equipped Macs running an OS with USB support. Fully compatible with the ADB version of MacHASP.



MacOS



USA: 1-800-223-4277
212-564-5678
Int'l: 972-3-6362222

www.aks.com/mt

ALADDIN®

KNOWLEDGE SYSTEMS LTD

Mac software protection provider, Micro Macro Technologies, becomes part of Aladdin, giving its customers even better service from the number one name.

By John C. Welch

MacNFS

Reviewing Thursby Unix Connectivity Package

BACKGROUND

This month we will take a look at MacNFS 3.0, from Thursby Systems. MacNFS is an NFS client that allows you to mount Unix NFS shared file systems on your Mac. This is a one way sharing, as MacNFS does not allow you to share your files to Unix users via NFS. With OS X looming large, and NFS being a key type of filesharing under that OS, NFS connectivity for the 'Classic' Mac OS is worth looking at.

MACNFS

MacNFS is a NFS, (Network File System) client for any Mac with at least a 68030 CPU, Mac OS 7.6 or higher, and Open Transport. To access Unix NFS shares, or file systems, the Unix box needs to be using NFS version 2 or greater, PCNFSD, (this is the daemon, or process that allows MacNFS to log in to the Unix box to mount drives) version 2 or greater, Network Lock Manager version 3 or greater, and Mount program version 1 or greater. For my tests, I ran MacNFS on a

variety of G3, G4, and other PowerMacs and PowerBook G3s. Unix servers were a mix of Sun boxes running Solaris 7, and SGI boxes running the current version of Irix, 6.5.X.

The installation of MacNFS is straightforward, with your options consisting of which parts of MacNFS to install. The entire package is a User's guide, the MacNFS Chooser extension, and online help, which consists of Mac Help files and an Apple Guide file. Once you install, you restart your Mac and serialize your copy. To initially use MacNFS, you open the Chooser, and select the MacNFS icon. You are then presented with a dialog requesting your name, company, and license number. This is my first minor beef with MacNFS, and Thursby in general. The license numbers that Thursby uses are some of the more complex ones I've seen in the Mac market. This by itself is not bad until you buy 30 copies of MacNFS, and get a page of labels, each with its own license number. A single license number for multiple copy purchases would be a good idea, especially if you remotely install MacNFS on multiple Macs at different locations.

Once you have entered the license number, you can then start setting up MacNFS. The initial presentation in the Chooser shows you a listing of all machines on your subnet that are exporting NFS shares. This is shown in Figure 1.

John Welch <jwelch@aer.com> is the Mac and PC Administrator for AER Inc., a weather and atmospheric science company in Cambridge, Mass. He has over fifteen years of experience at making computers work. His specialties are figuring out ways to make the Mac do what nobody thinks it can, and showing that the Mac is the superior administrative platform.

Mac Support Since 1985!

New Version!
Enhanced Security
Options, New
Platforms, More...

c-tree Plus® ONE EMBEDDED DATABASE TOOL THAT FITS ALL YOUR APPLICATIONS

FairCom has been providing fast, flexible and scalable database development tools to the commercial developer for over 20 years. During this time FairCom has been utilized within countless embedded appliances, web server development projects and many vertical market applications. By offering industry leading performance, unsurpassed multi-user data availability and a complete transaction enabled database Server, FairCom provides the depth and breadth of technology to fit all of your database development needs. Add FairCom to your toolbox today.

ONE FAST ISAM TOOL, MANY PLATFORMS

Every copy of c-tree Plus supports all these platforms: Mac OS, Mac OS X, MK Linux, Linux (Intel...), Novell Netware, OS/2, Solaris (SPARC), Solaris (Intel), Sun OS, AIX, HP UX, SCO, Interactive, AT&T Sys V, QNX, 88OPEN, FreeBSD, Windows 95/98/2000/NT, Lynx, Banyan Vines, and more...

Plus, support for ADSP, SPX, TCP/IP



Vertical Markets

APPLICATION	BENEFIT
Library Management	Fast
Insurance	Reliable
Inventory Control	Scalable
Point of Sale	Cross Platform

c-tree Plus®

- Industry leading performance
- Seamless cross platform migration
- Unsurpassed multi-user data availability
- Complete source code
- Portable threading API
- Thread safe libraries
- Low total cost of ownership!

FairCom® Server

- Powerful multithreaded database server
- Available for over 25 platforms
- Robust heterogenous network support
 - Mix and match clients and servers
- Multithreaded client support
- Full transaction processing
- Automatic file recovery
- Very small RAM and disk footprint
- Easy installation/configuration
 - No DBA required
- Much more!

...IN ONE CONVENIENT
PACKAGE FOR ONLY \$895

FairCom®

C O R P O R A T I O N

Commercial Database Solutions Since 1979

FairCom Offices

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802



www.faircom.com/mac • USA. 800.234.8180 • info@faircom.com

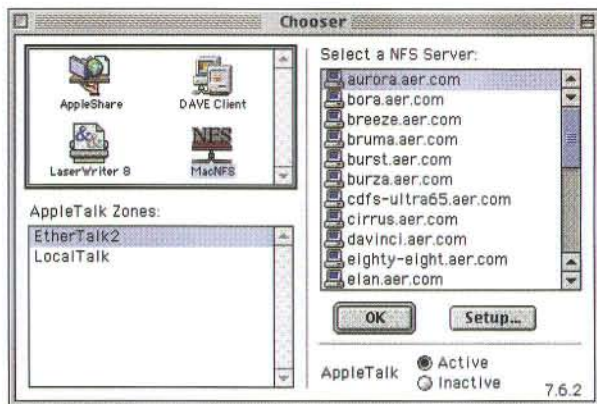


Figure 1. MacNFS Server List.

This list looks like any other server list you would see in the Chooser. You have the 'OK' button, which allows you to select the server you want to log into. The other button takes you into the MacNFS client setup. As you can see in **Figure 2**, the setup for MacNFS is fairly straightforward.

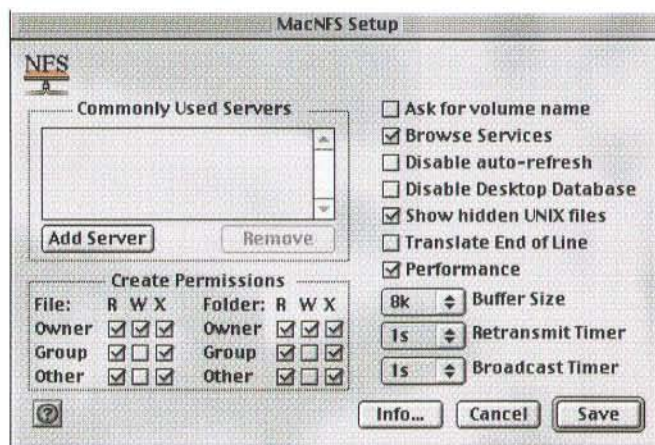


Figure 2. MacNFS Client Setup.

The options allow you to configure the basic behaviors of the client. For the most part, the default options will work. Most administrators will want to configure the create permissions, which control how the NFS filesystems apply file access rules to files created using the MacNFS client. The 'Show hidden UNIX files' option controls the visibility of Unix .files, which are normally configuration files, and not viewable by non-administrators. 'Translate End of Line' automatically handle converting Unix line-end characters to the Mac line ends. The 'Info...' button shows the statistics for the client, and allows you to change the license code for that client if needed. Once you have made the necessary changes, clicking 'Save' commits those changes, and returns you to the server list.

Picking a server to access disks from is similar to doing it via AppleShare. You select a server, and either double-click, or click 'OK'. You are then presented with a login screen, shown in **Figure 3**.

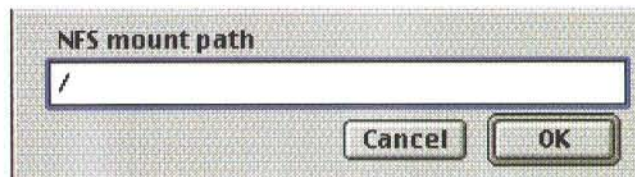


Figure 3. Login Screen.

The login has one unique feature, that being the 'Security:' pulldown menu. This is a listing of all visible servers that are running the PCNFSD authentication daemon. This allows the user to authenticate against different servers if the primary is down, or a specific set of NFS shares can only be authenticated by a specific server. Also, the user does not have to authenticate to the same server that they are trying to mount, so a single authentication server can be used for all NFS shares.

Once the user has authenticated, a list of default shares for that server is displayed. This screen, in **Figure 4**, is where the user selects the share(s) they wish to access, and to set share-specific options.

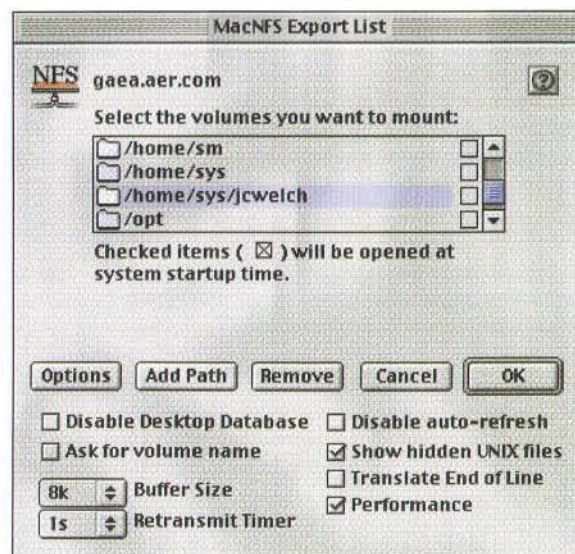


Figure 4. Default Shares.

This screen has many of the same options as the main settings screen, but any options set here, only apply to a specific share or shares. Like other network clients, the user can set certain shares to automatically mount on system startup. Unfortunately, this version of MacNFS does not allow for the KeyChain to be used to store the passwords for these shares. The 'Options' button toggles the options section of the screen on or off, (shown ON in **Figure 4**.) The 'Add Path' button brings up a dialog that allows the manual entry of a share that does not appear in the window. These shares must be entered using Unix path conventions, and the entire path must be entered manually. An ability to browse to the desired share would make this option easier to use, as Unix paths can be quite complicated. The 'Add Path' dialog is shown below in **Figure 5**.



Figure 5. Add Path Dialog.

Once the share has been selected and mounted, it behaves as any other network share would. Since Unix does not use dual streamed filesystems, MacNFS keeps the resource fork information in a separate file, that has the same filename as the data fork, but preceded by a percent sign. File access and usage is as fast as the network allows, but in general, it is speedy over 10Mbps Ethernet.

Thursby has done an excellent job in making MacNFS easy to use on a wide range of Macs. Although there are some minor issues with the manual path addition process, lack of KeyChain integration, and the serial numbering process, these are relatively minor quibbles, and are not bad enough to cause second thoughts about using MacNFS..



- no complicated reseller rebates
- no special terms or conditions
- no hidden charges
- no "get this if you buy this"
- no hooks, catches, installation fees, or well, any other bull!

PowerMax is all about choices. With our crack sales team always willing to consult and advise on everything Macintosh® combined with the ability to order on line, via email, fax or on the phone, we offer what you want, when you want, and at the price you need.

The G4 "Bulldozer" Bundle

New G4/400
Upgraded to 128 MB RAM • 10 GB HD
DVD-ROM • 56k Modem • 15" Apple LCD Display
Canon BJC-2100 inkjet
\$2099 after Canon printer rebate



The iMac™ "Bullseye" Bundle

Refurb Blueberry iMac 350
Upgraded to 128 MB RAM
6 GB HD
CD-ROM
56k Modem
Canon BJC-2100 inkjet
\$799 after Canon printer rebate



The iBook™ "Bullfighter" Bundle

Refurb Graphite iBook 366 MHz
Upgraded to 128 MB RAM
6 GB HD • CD-ROM
56k Modem • Graphite iBook carry case
• Canon BJC-85 portable inkjet
\$1699



The PowerBook™ "Bulldog" Bundle

Refurb PowerBook 500 MHz
Upgraded to 192 MB RAM
12 GB HD • DVD-ROM
56k Modem
Curtis Roadster Bag
Canon BJC-85 portable inkjet
\$2999



Examples of some of our great CPU deals:

Used 7100/66 16/700/CD	\$148
Used 7200/90 32/500/CD	\$159
Used 8100/80 40/500/CD	\$219
New Blueberry iMac™ 350 64/6/CD	\$749
Used 9600/200 64/2/CD	\$874
Ref Ruby iMac™ 450 DV 64/20/DVD	\$1088
Ref Indigo iMac™ 450 DV 64/20/DVD	\$1088
New Graphite iMac™ 400 DV 128/13/DVD	\$1188
Used iBook™ 300 32/3/CD	\$1199
Used Blue&White G3/400 64/6/DVD/56k	\$1249
Ref G4/450 128/20/DVD/Zip/56k	\$1749
And literally hundreds more!	\$Call

Just ask us about our great deals on:

- Iomega • Epson • Agfa • Canon • QPS
- Umax • Asante • Initio • Fantom • Lacie
- Kodak • Griffin • Sony • VST • PowerLogix
- ATI • Keyspan • Microtech • 100s more

TradeUp

603 604 601

G4 with PowerMax!

We'll take your Mac® OS computer in trade toward the purchase of new product. Call one of our expert Mac® consultants for full details!

800-689-8198
www.powermax.com

Local: (503) 624-1827 • Fax (503) 624-1635
email: sales@powermax.com

- Personal Financing • Fast P.O. Approvals • Business Leasing • Weekly specials on our web site



Knowledge is Power



Prices subject to change without notice. Prices reflect cash discount. Credit card orders strictly verified against fraudulent use. With use of credit card as payment customer acknowledges that some products are subject to final sale. Many prices are limited to stock on hand. All brand or product names are registered trademarks of their respective holders.

By *Éric Simenel*

Carbonization 101

*Easier than you might think
and so rewarding*

THE BENEFITS OF CARBON

A beta version of Mac OS X has just been released to the public and a final version should be released in early 2001.

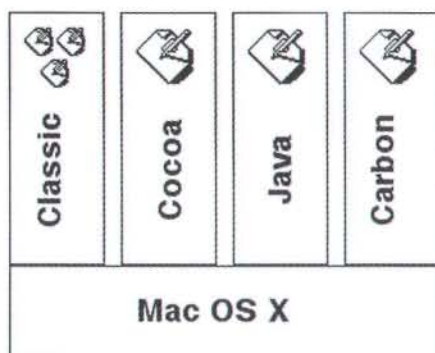


Figure 1. Mac OS X applications.

Four different kinds of applications, shown in **Figure 1**, can run on Mac OS X:

- 1 unmodified Mac OS applications currently running on Mac OS 9 or earlier, will run in the Classic environment. The User Experience will be the same as the one provided with Mac OS 9, and if one of the applications running in that environment encounters a serious problem, it may bring down the whole environment and thus any other application running in that environment.
- 2 Java applications, developed using only the JDK and companion libraries.
- 3 Cocoa applications, developed in either Java or Objective-C (possibly mixed with C/C++), and using the Cocoa frameworks.
- 4 and, last but not least, Carbon applications which are slightly modified Mac OS applications.

Each of the last three kinds of application, Java, Cocoa, and Carbon, all run as separate processes in their own protected memory area. If any of those applications encounters a serious problem, it won't affect the others.

The benefits, then, of Carbonizing an application should be apparent: for the cost of a few hours or days of work, you get a much more robust and impervious application that your users can run without problems. Another advantage is that a carbonized application gives you the Aqua theme user interface on Mac OS X.

Carbon being present, through the installation of CarbonLib, on Mac OS 9, 8.6, and 8.1, a Carbonized application, as a single binary, can run on all Mac OS from 8.1 to X.

THE SCOPE OF CARBON

Most applications and some plug-ins can be carbonized successfully. Extensions (INITs), Control Panels (cdev), and drivers of any sort cannot be carbonized since they don't run on Mac OS X and the main goal of Carbon is to provide an easy transition for developers from Mac OS 8 and 9 to Mac OS X.

Which version of Carbon should be used?

The version of Carbon that you choose depends on your goals and the configurations that you are targeting. If you want the maximum Mac OS coverage, use only Carbon 1.0.4 APIs which are supported from Mac OS 8.1 to Mac OS X. CarbonLib 1.0.4 gives most of the Toolbox APIs, plus the Control, Window, and Menu Properties, Navigation Services, the Core Foundation APIs, and Carbon Printing 1.0. If, when carbonizing, you find that

Éric Simenel simenel.e@apple.com is a software engineer in Apple's DTS (Developer Technical Support) team. He is currently supporting Mac OS 9 and Carbon.

// Like most Mac developers, // I easily spend 12 hours a day

// staring at line after line of C++ code in tiny, 9 point Monaco.
// Sometimes it makes my eyes feel like they're on fire.

```
{  
So the last thing I need is some  
fuzzy monitor that adds to my headaches.  
}
```

/***** begin excitement *****/

// That's why I'm so jazzed about this SGI monitor.

// Its ultra-high resolution = 1600 x 1024 and dpi = 110,
// giving me razor-sharp contrast.
// And the high refresh rate is
// perfect for poring through lines of code.

// At first, I was amazed at the clarity, the fine details that emerged.

```
{  
It was like seeing things for the first time.  
}
```

// Later, though, I learned to appreciate the wide aspect ratio [= 16:10],
// with a generous 17.3 inches of viewing area. SGI's 1600SW
// lets me have all my documents viewable at once, and it's
// a flat panel so it fits on my desk with room to spare.

// From the moment I saw this thing I was hooked. You will be, too.

```
{  
Especially when you find out  
how affordable it is.  
}
```

// Check it out.

/***** end excitement *****/



Michael Whittingham
V.P., Engineering
Wired, Inc.

www.sgi.com/flatpanel

This advertisement was written by an actual engineer. This is the first time in 7 months that he's been seen during the day.
© 2000 Silicon Graphics, Inc. All rights reserved. SGI 1600SW and SGI are registered trademarks of Silicon Graphics, Inc.
Mac is a registered trademark of Apple Computer, Inc. For more information, visit our website or call 1-888-SGI-7373.



sgi™

you are missing a desired functionality, check if it is provided in the Carbon 1.1 APIs which give you access to Appearance 1.1, Carbon Events (which offer an alternative to patching the Toolbox), the new DataBrowser control, ATSUI, URL Access, Keychain, FontSync, XML, HTMLRenderingLib, and others. Choosing the Carbon 1.1 APIs will restrict your target configurations to Mac OS 8.6 and above.

Note: Carbon on Mac OS X, and its companion CarbonLib for Mac OS 8 and 9, are not a closed API but an ongoing development. Future Apple technologies might be delivered only through the Carbon APIs.

THE CARBON PORTING PROCESS

Let's examine the nuts-and-bolts to see how porting to Carbon can be done. The goal of this article is not to be a rewrite of the "Carbon Porting" document which can be found in the Carbon SDK (see the bibliography at the end of the article for useful URLs), but rather a step-by-step guide based on DTS Engineers' experiences during various Carbon Porting lab events with 3rd-party developers at Apple.

This is not just a bunch of recipes, although you may find them very useful, but rather a methodology to apply when you encounter a problem not yet covered in any documentation.

Estimating how much work will be required

Of course, the amount of work depends on the size of the source code and the number of unsupported APIs in your source code. Experience in the Carbon Porting labs tells us that most developers can successfully carbonize 80% or more of their applications in the 3 to 5 days of that event.

CarbonDater is a tool which may help you quickly determine the amount of work to be done. An example of a Carbon Compatibility Report is shown in **Figure 2**.

Carbon Compatibility Report

Report for: TypeServicesPortUnicode
 File Creator: ????
 File Type: APPL
 Report date: 8/1/2000
 Data gathered by analyzer version: 1.3
 Data gathered on: 8/1/2000

Summary

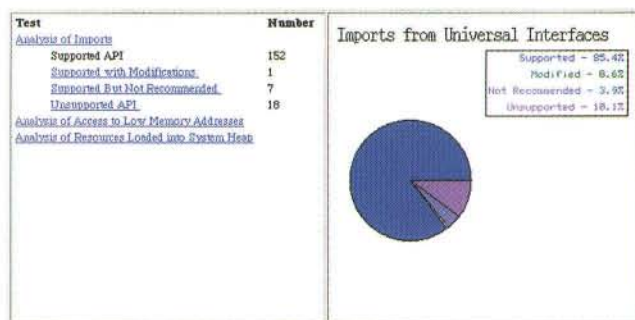


Figure 2. The Carbon Compatibility Report.

The Carbon Compatibility Report also provides a fairly detailed analysis and some suggestions as to how to replace unsupported APIs (**Figure 3**) and the kinds of modifications you will have to make in order to be Carbon-compliant (**Figure 4**).

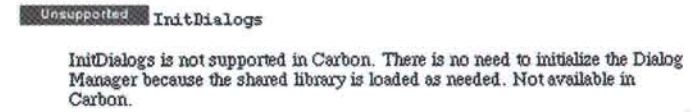


Figure 3. Unsupported API.

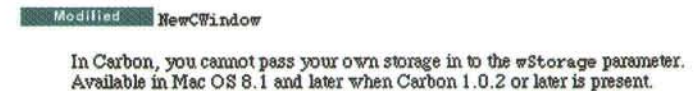


Figure 4. Modified API.

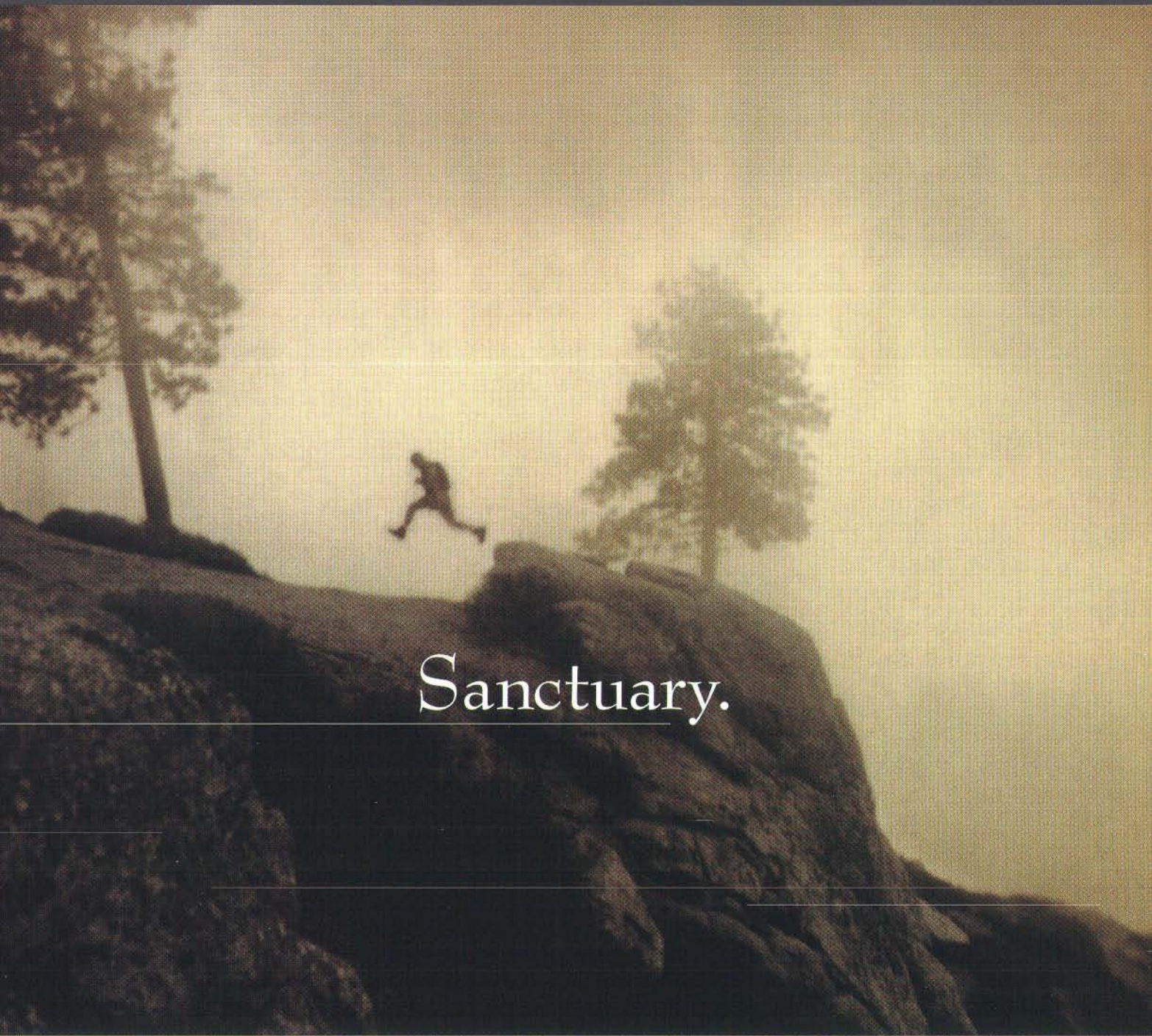
Another way to estimate the necessary amount of time is to plunge right into the Carbonization process; the first compilation will give you an idea of the extent of the work to be done.

First things first

As the "Carbon Porting Guide" suggests, it's best to update your project to the latest Universal Headers first. If you're interested in Carbon 1.0, then use the Universal Headers 3.3.2, but if you're interested in Carbon 1.1, then use the Universal Headers 3.4. If you are developing with CodeWarrior, be careful to follow the instructions (from your CodeWarrior User's manual) and update any pre-compiled headers if you're using them.

The next step depends on the size of your project and the way you prefer to develop. You can either follow all the steps described in the "Carbon Porting Guide" (which is preferable if your project is large), or follow these instructions (which assume you are using CodeWarrior, any version after and including Pro 2):

- first, clone your existing PowerPC target,
- second, add the preprocessor command `#define TARGET_API_MAC_CARBON 1` to the beginning of your project's prefix file (if you already have a prefix file, then duplicate it, add the preprocessor command in one of them and use this one for your Carbon target),
- and third, remove all Apple stub libraries such as InterfaceLib and any others that you might have needed in your project such as AppearanceLib, WindowsLib, etc., and add the CarbonLib stub instead. You can keep non-Apple libraries such as "MSL RuntimePPC.Lib", "MSL C.PPC.Lib", etc.



Sanctuary.

Move to **digital.forest**. The leading Internet Service Provider for the Macintosh community.

You need maximum flexibility, security and performance from your ISP. You'll find it with digital.forest. We're the proven provider of Macintosh Internet service.

Founded in 1994, our specialty is Mac systems and technologies. In fact, digital.forest is the world's largest Macintosh Hosting Provider. And our hard-earned knowledge of Mac server software is the broadest in the business.

We pioneered Macintosh Server Colocation and FileMaker Pro database hosting services. And our commitment to service keeps growing. So whether you choose us to host your files or house your servers, you've chosen the best.

Go with the industry leader. Discover the sanctuary of digital.forest.

digital.forest



8 7 7 - 7 2 0 - 0 4 8 3

info@forest.net
www.forest.net

Call toll-free in the U.S. and Canada.
For international inquiries, please call 425-483-0483.



FileMaker Pro is a registered trademark of FileMaker, Inc. The digital.forest logo is trademark or registered trademark of digital.forest, Inc.

© digital.forest, Inc. 2000. All rights reserved

If you are using any other libraries, then you will need to determine (the PEF Viewer utility can help you do that) if the code they contain is importing any symbols from InterfaceLib or any of those Apple libraries at runtime. If that's not the case then they should be safe to use as is, but if they are, then those libraries will have to be carbonized as well.

After you're finished with those 3 steps, which shouldn't take you more than a few minutes, then attempt your first build.

You will get many, perhaps thousands, of errors.

Luckily, 80% of them, if not more, are repeats and just illegal access to Toolbox structures which are now opaque or illegal type usage (DialogPtr, WindowPtr, and GrafPtr are now 3 different pointer types and can't be used in place of each other). Addressing those errors is simple but tedious, and it just takes time. There are 2 different ways to fix those errors, one involving the CarbonAccessors.o library, and the other one involving conditional compilation. For instance, the following extract, which should be quite common in your source code:

```
WindowPtr theWind;
...
SetPort(theWind);
```

is going to generate the compiler error:

```
Error:
cannot convert 'OpaqueWindowPtr *' to 'OpaqueGrafPtr
*'
```

You can either change the source code to

```
SetPortWindowPort(theWind);
```

and add the CarbonAccessors.o library to your non-Carbon targets in your project or change the source code to

```
#if !TARGET_API_MAC_CARBON
    SetPort(theWind);
#else
    SetPortWindowPort(theWind);
#endif
```

and you don't have to add the library.

The advantage of the former solution is that the source code is more readable but your non-Carbon application is going to grow by about 60K, whereas the advantage of the latter is that your non-Carbon application stays rigorously the same, but you will have to add a lot of those `#if !TARGET_API_MAC_CARBON` lines in your source code which can also look like:

```
#if !TARGET_API_MAC_CARBON
    yLocation = theWind->portRect.bottom-10;
#else
    Rect thePortRect;
    GetWindowPortBounds(theWind, &thePortRect);
    yLocation = thePortRect.bottom-10;
#endif
```

It's your choice.

Finding out Carbon information

The best way to find out is to apply the following methodology: in the previous example, you tried to access the `portRect` field of the `WindowPtr` `theWind`. This is doubly illegal because the variable `theWind` is a `WindowPtr` and not a `GrafPtr`, and also because both `WindowPtr` and `GrafPtr` are opaque structures anyway. So the obvious solution is to look at the `GrafPort` structure definition in `QuickDraw.h` and there you will see that it changed to:

```
struct GrafPort {
    short device; /* not available in Carbon */
    BitMap portBits; /* in Carbon use
GetPortBitMapForCopyBits
or IsPortColor */
    Rect portRect; /* in Carbon use Get/SetPortBounds */
};
```

and if you look for `GetPortBounds` you will find it at the end of `QuickDraw.h` in the `/* Getters */` section:

```
EXTERN_API( Rect *)
    GetPortBounds(CGrafPtr port, Rect * rect);
```

So now you know how to get the `portRect` out of a `GrafPtr` (or a `CGrafPtr`, both are equivalent in Carbon since the structures they point to are opaque), but what you have is a `WindowPtr`. Again, since the port of the window is in fact a field in the `WindowRecord` structure you just open `MacWindows.h` and you look at the `WindowRecord` (or `CWindowRecord`) structure definition and there you will see that it changed to:

```
struct WindowRecord {
    GrafPort port; /* in Carbon use GetWindowPort */
    short windowKind; /* in Carbon use Get/SetWindowKind */
};
```

And if you look for `GetWindowPort` you will find it at the end of `MacWindows.h` in the `/* Getters */` section (do you see a pattern here?):

```
EXTERN_API( CGrafPtr ) GetWindowPort(WindowRef
window);
```

Don't worry about the `WindowRef` type which is the same as `WindowPtr` (defined in `QuickDraw.h` for practical reasons).

So you could write:

```
#if !TARGET_API_MAC_CARBON
    yLocation = theWind->portRect.bottom-10;
#else
    Rect thePortRect;
    CGrafPtr theWindowPort;
    theWindowPort = GetWindowPort(theWind);
    GetPortBounds(theWindowPort, &thePortRect);
    yLocation = thePortRect.bottom-10;
#endif
```

but, in order to simplify and reduce the coding, Apple also provided a direct `GetWindowPortBounds` call that you also find at the end of `MacWindows.h`:

```
EXTERN_API( Rect *)
    GetWindowPortBounds(WindowRef window, Rect *
bounds);
```

so that's why you can write the shorter code shown in the previous section. Apple provided those shortcuts for the most common usage.

Since the `GetWindowPortBounds` call (like the others) returns the same `Rect*` as being passed as the second parameter, you could even write:

```
#if !TARGET_API_MAC_CARBON
    yLocation = theWind->portRect.bottom-10;
#else
    Rect thePortRect;
    yLocation =
        GetWindowPortBounds(theWind, &thePortRect)-
        >bottom-10;
#endif
```

or even, if you keep a global scratch variable `Rect gTempRect` around which will be used for such operations you could write:

```
#if !TARGET_API_MAC_CARBON
    yLocation = theWind->portRect.bottom-10;
#else
    yLocation =
        GetWindowPortBounds(theWind, &gTempRect)->bottom-
        10;
#endif
```

Again, it's your choice.

Sometimes, the comments provided in the Header files for Carbonization advice are not on the same line as the information you're looking for, but usually they're not too far away. In the following case for example:

```
GrafPtr currentPort;
GetPort(&currentPort);
EraseRect(&currentPort->portRect);
InvalRect(&currentPort->portRect);
```

will bring you:

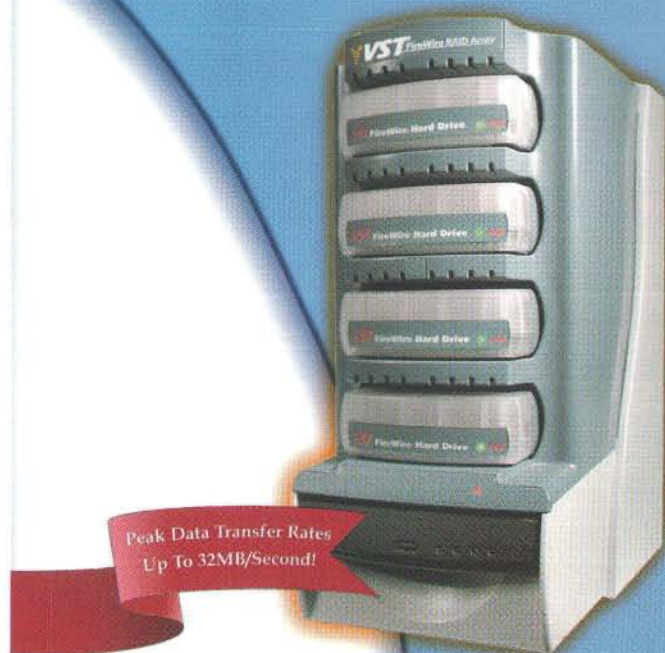
```
Error : undefined identifier 'portRect'
SampleWindows.cp line 336 EraseRect(&currentPort-
>portRect);
Error : undefined identifier 'InvalRect'
SampleWindows.cp line 337 InvalRect(&currentPort-
>portRect);
```

The incorrect usage of `portRect` will be fixed by a simple call to `GetPortBounds` as previously but what about `InvalRect`? A search for `InvalRect` (in `MacWindows.h`) will make apparent that we now have:

```
#if CALL_NOT_IN_CARBON
EXTERN_API( void ) InvalRect(const Rect * badRect)
ONEWORDINLINE(0xA928);
...
#endif /* CALL_NOT_IN_CARBON */
```

so we have to look around and we will see that for this call as well as for `InvalRgn`, `ValidRect`, and `ValidRgn`, we have a comment above which says:

```
/*
These aren't present in Carbon. Please use the InvalWindowRect, etc.
routines
below instead.
*/
```



FireWire with a DIFFERENCE

VST's line of portable USB, FireWire® and PowerBook® peripherals represent the ultimate in performance, ultra-compact, light-weight packaging, and plug 'n' play ease of use.

Taking FireWire beyond the limits

VST's New FireWire RAID Array packs 120GB into the smallest possible space, supports RAID levels 0 and 1 simultaneously, and even operates on a PowerBook Li-Ion battery for up to three hours of on-location use.

VST's New FireWire RAID Array is a powerful tool for demanding professional applications, such as video and audio broadcasting, where large amounts of graphics, digital video and music must be captured and manipulated in real time for storage or editing. For performance and high capacity in FireWire storage VST FireRAID™ is the solution!

Visit us at www.vsttech.com to learn more about this and other products designed to simplify the digital lifestyle.

FireWire Hard Drive
Our amazing, FireWire hard drive provides the world's smallest, fastest, portable drive with capacities from 1GB to 30GB.

FireWire/USB Hard Drive
Our combo drive includes FireWire & USB ports for maximum compatibility with capacities up to 30GB.

Full Height FireWire Hard Drive
With sizes available up to 75GB, the Full Height FireWire Hard Drive is the ideal solution for those looking for high speed desktop storage.

UltraTide66
An instant RAID solution for your G4 or G3. SoftRAID provides up to 66MB/sec data transfer with 60, 90 & 150GB of storage.



So we will diligently replace the illegal code with:

```
Rect thePortRect;
GetPortBounds(currentPort, &thePortRect);
EraseRect(&thePortRect);
InvalWindowRect(GetWindowFromPort(currentPort),
                &thePortRect);
```

You will note that we also used the utility function `GetWindowFromPort` which you can find in `MacWindows.h` as well.

What happened to my qd?

The global variable `qd` has been removed so if you had been using some code like:

```
DragWindow(theWind, theEvent->where,
           &qd.screenBits.bounds);
```

then, after looking for `screenBits` in `QuickDraw.h` and finding the relevant comment, you will have to write:

```
Rect bounds;
#if !TARGET_API_MAC_CARBON
    bounds = qd.screenBits.bounds;
#else
    BitMap bmap;
    GetQDGlobalsScreenBits(&bmap);
    bounds = bmap.bounds;
#endif
DragWindow(theWind, theEvent->where, &bounds);
```

What happened to my InitGraf and other Toolbox Managers Initialization calls?

Most of them are now just unneeded in Carbon so you can just conditional-compile out `InitGraf`, `InitFonts`, `InitWindows`, `InitMenus`, `TEInit`, and `InitDialogs`. You must still use `InitCursor`. Typical initialization memory schemes such as the manipulation of the heap limit (`SetApplLimit` and `GetApplLimit`) as well as `MaxApplZone` are also prohibited in Carbon.

For obsolescence reasons, the support for Desk Accessories is no longer required in applications so `SystemClick` and `OpenDeskAcc` have also been removed from Carbon.

So is it just searching, finding, and replacing?

Although at least 80% or more of the errors you will get at the first compilation will indeed be no more problematic than that, still it's not going to be that simple for the remaining 20% or less.

First you have to deal with the modified calls such as `NewCWindow` (and `NewWindow`, `NewDialog`, and `NewColorDialog` as well for the same reason) which does not accept a storage parameter anymore. Unfortunately the compiler will not alert you that such calls are going to fail although you will get a null return value from the call so if you have a good error-checking code you will easily detect the problem at runtime rather than crashing. The main reason developers were passing a storage as first parameter

is that they wished to extend the Toolbox structure with their own fields, and to lessen heap fragmentation. Since those structures are opaque, this is no longer possible. So you will have to modify such declarations:

Note: All error-checking code has been removed from the sample code extracts present in this article so that the modifications you have to apply in order to Carbonize are more evident. Real code would need error-checking...

```
typedef struct
{
    WindowRecord theWind;
    OSType theSignature;
    GWorldPtr theGWorld;
    ...
} MyWind, *MyWindPtr;
```

in

```
typedef struct
{
    #if ! TARGET_API_MAC_CARBON
        WindowRecord theWind;
    #else
        WindowPtr theWind;
    #endif
    OSType theSignature;
    GWorldPtr theGWorld;
    ...
} MyWind, *MyWindPtr;
```

and change from:

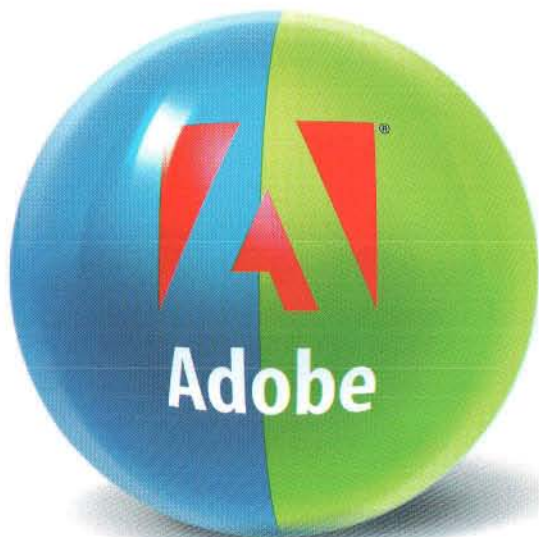
```
wind = (MyWindPtr)NewPtr(sizeof(MyWind));
if (wind != 0L)
{
    wind->theSignature = kSig;
    NewCWindow(wind, &rBounds, fName, 1,
               noGrowDocProc, (WindowPtr)-1L, 1, 0L);
    ...
}
```

to:

```
wind = (MyWindPtr)NewPtr(sizeof(MyWind));
if (wind != 0L)
{
    wind->theSignature = kSig;
    #if ! TARGET_API_MAC_CARBON
        NewCWindow(wind, &rBounds, fName, 1,
                   noGrowDocProc, (WindowPtr)-1L, 1, 0L);
    #else
        wind->theWind = NewCWindow(NULL, &rBounds, fName,
                                     1,
                                     noGrowDocProc, (WindowPtr)-1L, 1,
                                     (long)wind);
    #endif
    ...
}
```

and you can still extract your `MyWind` structure location from the `refCon` value of your `WindowPtr`. If you were already using the `refCon` field to store some other information, then you can keep that information in the structure `MyWind` instead. Another solution is to use the new functions `SetWindowProperty` and `GetWindowProperty` which enable to attach tagged information to any window.

Another likely modification you will have to make is the control list walk-through. Previously you could just start from the `controlList` field of the structure `WindowRecord`, but of course this



Some companies are really on the ball.



Some of the world's most forward-thinking companies have transformed their Web sites into live meeting rooms. They have discovered the speed, reliability and scalability of the WebEx Interactive Network. Their Web sites now hum with people conducting business right in their browsers, without any hardware or software changes. Now you can give presentations. Share software and desktops. Tour the Web. Voice and video conference.

All in real-time. All on the Web. Talk about ROI. Isn't it time your company got on the ball? Visit webex.com or call 1-877-50-WebEx to see how great minds meet online.

great minds meet online at webex.com



field is no longer accessible in the opaque structure and there are no accessor function to retrieve it. In order to get the first control of the control list of a window, you will have to modify your code like this:

```
// window creation with NewWindow, or NewCWindow,
// or CreateNewWindow (preferred...), or...

#if TARGET_API_MAC_CARBON
    ControlRef theRootControl;
    if (theWindow != NULL)
        CreateRootControl(theWindow, &theRootControl);
#endif
```

and later:

```
#if !TARGET_API_MAC_CARBON
    ControlHandle theControl = (ControlHandle)
        (((WindowPeek)theWindow)->controlList);
    while (theControl != NULL)
    #else
        UInt16 numChildren, index;
        ControlRef rootControl, theControl;
        GetRootControl(theWindow, &rootControl);
        CountSubControls(rootControl, &numChildren);
        for (index = numChildren; index >= 1; index--)
    #endif
    {
        #if TARGET_API_MAC_CARBON
            GetIndexedSubControl(rootControl, index,
                &theControl);
        #endif
        // do something with theControl,
        // you can even dispose it since we're doing a reverse loop
        #if !TARGET_API_MAC_CARBON
        #if DISPOSINGCONTROL
            theControl = (ControlHandle)
                (((WindowPeek)theWindow)->controlList);
        #else
            theControl = (*theControl)->nextControl;
        #endif
        #endif
    }
```

If you didn't add support for the Navigation Services, to replace the use of the Standard File Package, now is the time since Carbon does not support the Standard File Package.

Another whole section of the traditional Macintosh Toolbox, the Scrap Manager, has also been revised in Carbon. Most applications use the Scrap Manager routines in a fairly straightforward way and the amount of modifications you may have to apply may not be much more complicated than the following.

For copying:

```
#if !TARGET_API_MAC_CARBON
    ZeroScrap();
#else
    ScrapRef scrapRef;
    ClearCurrentScrap();
    status = GetCurrentScrap(&scrapRef);
#endif

#if !TARGET_API_MAC_CARBON
    PutScrap(byteLength, 'utxt', *hUnicodeText);
#else
    status = PutScrapFlavor(scrapRef, 'utxt',
        kScrapFlavorMaskNone, byteLength, *hUnicodeText);
#endif
```

And for pasting:

```
#if !TARGET_API_MAC_CARBON
    LoadScrap();
#else
    ScrapRef scrapRef;
    status = GetCurrentScrap(&scrapRef);
#endif

#if !TARGET_API_MAC_CARBON
    long offset, byteLength = GetScrap(NULL, 'utxt',
        &offset);
#else
    long byteLength;
    status = GetScrapFlavorSize(scrapRef, 'utxt',
        &byteLength);
#endif

    if (byteLength <= 0) return;

#if !TARGET_API_MAC_CARBON
    Handle hUnicodeText = NewHandle(0);
    GetScrap(hUnicodeText, 'utxt', &offset);
#endif

    // let's grab the text from the scrap
    UniCharCount uTextLength = byteLength /
        sizeof(UniChar);
    UniCharArrayPtr theUnicodeText =
        (UniCharArrayPtr)NewPtr(byteLength);

#if !TARGET_API_MAC_CARBON
    HLock(hUnicodeText);
    BlockMoveData(*hUnicodeText, (Ptr)theUnicodeText,
        byteLength);
#else
    status = GetScrapFlavorData(scrapRef, 'utxt',
        &byteLength,
        theUnicodeText);
#endif
```

Almost there since we should just comment out all the printing code for now. The Printing Manager has been revised also in Carbon and in such ways that it could be the only subject of another MacTech article by itself, so we're not going to deal with that in this article.

Patience and labor will get you to a point where you no longer get any compilation errors.

But now you'll see link errors.

Maybe some will be due to Toolbox calls present in the headers but missing in the CarbonLib shared library (this is still a work in progress), but most of the errors will come from the use of MSL (MetroWerks Standard Library). You will either have to use the updated for Carbon version of MSL from MetroWerks (which is part of CodeWarrior Pro 6 which also contains a fully carbonized version of PowerPlant) or get rid entirely of any dependencies on this package.

Again, patience and labor will get you to a point where you no longer get any link errors.

And you should thus be able to test your application on Mac OS 9 (recommended for the first runs), and it should run fine although you may have to increase the memory partition (resource 'SIZE' id -1).

Depending on the version of CarbonLib you've been developing against, you can also test the running of your application on 8.6 and 8.1.

But the real test will of course be the running on Mac OS X!

RUNNING ON MAC OS X

If you just launch your recently carbonized application on Mac OS X, you may not see a single difference in either behavior or even appearance with your application running on Mac OS 9. The most probable cause is that you're in fact running your carbonized application in the Classic environment! If you don't see your application interface using the Aqua theme user interface, then you are running in the Classic environment. In order for your application to be run as a Carbon Application, and thus in its own memory-protected space and with the Aqua theme user interface, you have to let Mac OS X know about the fact that you did your job and carbonized this application. You do that by adding both a 'carb' resource with id 0, the content does not matter, and usually you'll just set a single byte as 0, and adding a 'plst' resource, also with id 0, which contains the XML representation of the application's property list. If those 2 resources are present in your application then Mac OS X will launch your application as a Carbon process with all the advantages that come with it.

Note: If you have been following the suggested steps described above, you should have at least 3 different targets for your project: debug-Classic, non-debug-Classic, non-debug-Carbon. You may have more... Since all those targets can run on Mac OS X, either being launched in the Classic environment, or launched as a Carbon process, it might be useful to be able to identify at runtime, with certainty, which target you're

currently running. You could use the following routines (or adapt them to your needs) to achieve identification:

```

Boolean IsThisSX()
{
    long response;
    OSErr err = Gestalt(gestaltSystemVersion,
&response);
    return ((err == noErr) && (response >=
0x00000A00));
}

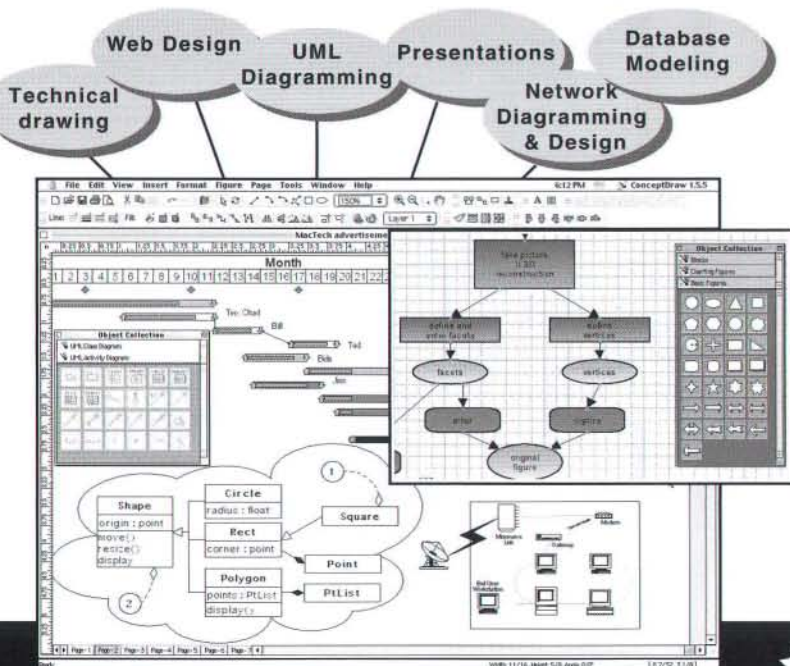
Boolean IsThisAqua()
{
    long response;
    OSErr err = Gestalt(gestaltMenuMgrAttr, &response);
    return
    (
        (err == noErr) &&
        ((response & gestaltMenuMgrAquaLayoutMask) !=
0)
    );
}

void DoAboutBox()
{
    Str255 theStr, theStr2, theStr3;
    #if __option (sym)
    sprintf((char *)theStr2, "Debug Version");
    #else
    sprintf((char *)theStr2, "Final Version");
    #endif
    #if TARGET_API_MAC_CARBON
    if (IsThisAqua())
        sprintf((char *)theStr3, "Carbon Aqua");
    else
        sprintf((char *)theStr3, "Carbon Platinum");
    #else

```



YOUR CROSS-PLATFORM TOOL FOR FLOWCHARTING AND TECHNICAL DRAWING
Powerful & Quick, Intuitive & Easy, Web-Oriented & Versatile, with Rich Customizable Libraries



★ Useful for any purpose

- 100% Mac/PC compatible
- Over 1700 objects in more than 80 task-related, expandable libraries
- Customizable grid, Snap & Glue tools, scale support for most precise drawing
- HTML export and hyperlinks let you create Web-ready documents
- Support for most popular graphic formats
- Smart reshapable connectors
- Intelligent objects with programmable behavior
- Support for layers and multiple pages
- Multi-level stack Undo
- Intuitive, user-friendly interface.

and so much more...

Download the FREE demo and see yourself!

Prices from..... **\$125**

★ Order your copy at www.ConceptDraw.com
or call for dealers near you: (800) 531-3227

★ Product of Computer Systems Odessa Corp.

```

    sprintf((char *)theStr3, "Classic");
#endif
    sprintf((char *)&theStr[1], "%s, %s, %s, %s",
        (char *)theStr2, (char *)theStr3, __DATE__, __TIME__);
    theStr[0] = strlen((char *)&theStr[1]);
    ParamText(theStr, "\p", "\p", "\p"); Alert(128, 0L);
}

```

When running on Mac OS X, you will notice that some system menus have changed and some menu items have moved. One important change is that the Quit menu item is now in the Application menu (which replaces the Apple menu of Mac OS 9 and previous). You are no longer supposed to have a Quit menu item in your File menu. That also means, by the way, that you need to have a **QuitAppleEventHandler** routine since this is the way that Mac OS X will inform your application that the user wants to quit it. You are not supposed to add the "desk accessories" (which have not been true desk accessories for a very long time anyway) in the Apple menu either.

So your menu initialization and installation routine should look like:

```

Handle menuBar = GetNewMBar(kMainMenuBar);
SetMenuBar(menuBar);
DisposeHandle(menuBar);
#ifdef TARGET_API_MAC_CARBON
    if (IsThisAqua()) // removing the Quit menu item and the
        separator line
    {
        MenuHandle menu = GetMenuHandle(kFileMenu);
        DeleteMenuItem(menu, kQuitMenuItem);
        DeleteMenuItem(menu, kQuitMenuItem-1);
    }
#else
    AppendResMenu(GetMenuHandle(kAppleMenu), 'DRVR');
#endif
DrawMenuBar();

```

And with that fine-tuning achieved, you are pretty much done. You will still have to thoroughly test your application on all Mac OS versions it's supposed to run on, but the job is no more complicated than described here (with the exception of printing which has to be rewritten). You may encounter some mysterious behaviors and the most frequently reported is that the drawing in one of your windows is not happening anymore (typically some kind of animation). The most likely reason for that is that you are drawing in that window outside of its update routine and since windows are double-buffered on Mac OS X, you just keep modifying the bits of the offscreen and nothing happens in the window on the screen. Carbon moves the bits from the offscreen to the window at the end of the update routine and in some other conditions, but not in that case, so you will have to add a bit of tweaking there as well. There are 2 ways to achieve that tweaking, but only one is recommended. **QuickDraw.h** provides 3 new APIs (**QDIsPortBuffered**, **QDIsPortBufferDirty**, and **QDFlushPortBuffer**) which can be used that way:

```

InitTheCircle();
for (i=startingMonth; i<=endingMonth; i++)
{
    UnsignedWide startingTime, endingTime;
    Microseconds(&startingTime);
    UpdateTheCircle(i);
}

```

/ on Mac OS X, drawing is double-buffered by default*/*

```

#ifdef TARGET_API_MAC_CARBON
    if (QDIsPortBuffered((CGrafPtr)curPort))
        QDFlushPortBuffer((CGrafPtr)curPort, NULL);
#endif
    do {Microseconds(&endingTime);} while
        (ComputeMicros(&startingTime, &endingTime) <
            kAnimationDelay);
}
TermTheCircle();

```

The other way, which is not recommended, is to add the **kWindowNoBufferingAttribute** attribute to the **WindowAttributes** parameter you pass to **CreateNewWindow**. In that case, the window will not be double-buffered so all drawing will be directed to the window, but you disrupt the normal updating mechanism of the Window Server and, for instance, if you move such a window, you may leave undesirable graphical artifacts on your screen.

CONCLUSION

If you have more than one application to carbonize, we advise you to start with the most recently developed first since it will be the easiest to carbonize. It should already be using a recent version of the Universal Headers and shouldn't do too many of the various no-no's no longer available in Carbon. Going back in time, carbonizing older and older applications, you will find that the previous experiences will help you to overcome the difficulties arising from carbonizing old code. But the methodology remains the same:

- 1) use at least the Universal Headers 3.3.2
- 2) for each error, refer to that routine or that structure in the appropriate header and read the Carbonization comments which are sure to be found around.

If for any reason you are at a loss as to what to do or how to do it, then you're more than welcome to join and participate in the Carbon Development mailing list that you can join by sending an email to <requests@sam.apple.com> with "subscribe carbon_development" as the email's message. The Carbon Development mail list is restricted to discussion of Carbon development issues.

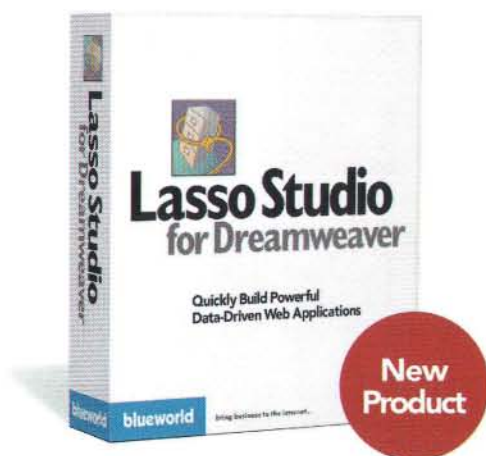
If you still can't find satisfaction when participating in the list, you can address your question to <ds@apple.com> but you need to be a member of our programs in order to do that.

REFERENCES

- CarbonDater: <http://developer.apple.com/macosx/carbon/dater.html>
- Carbon SDK: <http://developer.apple.com/sdk/>
- Documentation: <http://developer.apple.com/macosx/carbon/>
- Documentation: <http://developer.apple.com/techpubs/carbon/>
- Documentation: <http://developer.apple.com/technotes/tn/tn2003.html> "Moving Your Code to Mac OS X"

MT

Quickly Build and Deploy Powerful Data-Driven Web Applications



Lasso Studio and Lasso Web Data Engine™ Lead The Way.

Building custom and shrink-wrapped database-driven Web applications requires a whole new way of doing things. Having pioneered the Web Data Engine™ over three years ago, Blue World and the Lasso Web Data Engine consistently lead the way providing a feature set Web developers describe as "incredible." Build online stores, discussion forums, resource management systems and other demanding database-driven Web applications with unrivaled performance, ease, security, extensibility, control and flexibility. Develop using multiple languages—including LDML, CDML, Server-Side JavaScript, Java and XML—and deploy across multiple platforms. Your Lasso code works identically regardless to which database you're connected. Lasso solutions for FileMaker® Pro databases easily scale to big iron ODBC-compliant databases like Oracle, Informix, Sybase and more with little or no change. What's more, the Lasso Java Application Programming Interface (LJAPI) provides developers an easy-to-use Java-based API for unprecedented extensibility.

Find out why hundreds of thousands of websites rely on award-winning Lasso technology for their business critical Web data. Download a 30-day evaluation copy at www.blueworld.com/download/ or order securely online today at the Blue World Store at store.blueworld.com.

Lasso Product Line – The leading Web tools for Macintosh and beyond.

bring business to the internet.™

blueworld

by Sam Krishna and Patrick Taylor

Throughout this series we've asserted that programming in WebObjects is a richer, more mature experience than any competing web application development environment. A major reason for this superior programming experience can be laid at the feet of EOKeyValueCoding.

EOKeyValueCoding is simultaneously a powerful protocol (in Objective C) and interface (in Java). When the EOControl framework is imported, all Objective C classes that inherit from NSObject receive EOKeyValueCoding behavior. (The EOControl framework is imported automatically into WebObjects apps and frameworks. You do not have to explicitly import it in to your WebObjects app or framework project.) In Java, when the developer creates a subclass of EOEnterpriseObject, EOCustomObject, EOGenericRecord or WOComponent, then EOKeyValueCoding is available.

What is remarkable about EOKeyValueCoding is how universally powerful it is. With this protocol/interface, the WebObjects and EOF frameworks know how to access custom instance variables (ivars) from within your subclass. Using the methods valueForKey() and valueForKey() (in Objective C valueForKey: and valueForKey:) WOF and EOF can access your ivars through standard API. In addition, EOKeyValueCoding provides the bridge within EOF for effective entity-relationship mapping.

In valueForKey(), this standard method allows a lookup of an ivar from both WOF and EOF classes. Consider a WOComponent subclass that needs to fill in the value of a String object in the dynamic WOString element. The WOString is bound to lastName of an EnterpriseObject (EO) which represent the employees of a corporation.

ORDER OF ACCESS

1. The component subclass searches for a public accessor method based on the key name. With a key of "lastName", valueForKey() looks for a method named getLastName() or lastName(). In Objective C, it looks for a method named getLastName or -lastName.
2. If a public accessor method isn't found, the component subclass searches for a private accessor method based on key. Note that traditionally in the Apple frameworks, a private method is distinguished by a preceding underbar. In this case, valueForKey() looks for a method named

_getLastName() or _lastName(). In Objective C, it looks for a method named _getLastName or -_lastName.

3. If an accessor method isn't found and the class method accessInstanceVariablesDirectly returns true (in Java) or YES (in Objective C), valueForKey() searches for an ivar based on the key name and returns its value directly. For the key "lastName", this would be _lastName or lastName.
4. If neither an accessor method or an ivar is found, the default implementation invokes handleQueryWithUnboundKey() in Java or handleQueryWithUnboundKey: in Objective C. The default implementation of this method will raise an NSRangeException.

Sometimes exceptions are raised because although ivars have changed or been deleted for some reason the component subclass or the frameworks still expect the ivar to exist unchanged. In these situations after 10-15 minutes of proverbial pounding sand, we would override handleQueryWithUnboundKey() and check to see if the disputed ivar is being looked up, and if so, return nil or null.

EOKEYVALUECODING API TO ACCESS INSTANCE VARIABLES

For another example, suppose that you have an EO using a BigDecimal or NSDecimalNumber with "someDecimal" as one of its ivars.

Java

```
(BigDecimal)myEO.valueForKey("someDecimal");
```

note: you must always downcast your objects when using valueForKey() in Java

Objective C

```
[myEO valueForKey:@"someDecimal"];
```

Here is the operational order for valueForKey() (takeValueForKey: in Objective C)

1. The EOKeyValueCoding methods search for a public accessor method of the form setKey (setKey: in Objective C)
2. If a public accessor method isn't found, EOKeyValueCoding searches for a private accessor method of the form _setKey (Objective C _setKey:) invoking it if such a method exists

Please feel free to contact the authors with questions or comments on the articles at webobjectseof@mac.com . We may not be able to reply personally to all emails but every one will be read.

WHAT ARE PROTOCOLS AND INTERFACES?

Objective-C protocols and Java interfaces represent essentially the same object-oriented concept: the ability to declare methods that are independent of a particular class and can be implemented by classes in any hierarchy. Any Objective-C class that declares itself to conform to a particular protocol or Java class that implements a particular interface must, by default, implement the methods declared by that protocol/interface. The approaches taken are different with the Objective C approach being arguably more flexible.

Confused? Don't worry — maybe an analogy can make things clearer. Imagine you're writing a nature simulation program which requires that various "animals" have a swimming behavior. Since entirely different kinds of animals can share behaviors even if they come from entirely different species, phyla or families, not every swimming creature inherits from the same class (whales, trout, dogs and humans are all "swimmers" even though they inherit from different classes of "animal": whales do not inherit from trout because whales are mammals while trout are fish, dogs do not inherit from humans because dogs are canines while humans are primates, etc).

In your swimming simulation, you need to have your objects implement a specific type of behavior, but not necessarily the same way. You also want to show what happens when certain types of animals are not able to swim in water (like most species of birds, for example). So, if you are writing an Objective C program, you declare a

'Swimming' protocol. Correspondingly, if you are writing a Java program, you declare a 'Swimming' interface.

The Objective-C protocol declaration may look like this:

```
@protocol Swimming
- (void)treadWater;
- (void)swimFreestyle;
- (void)breathe;
....
@end
```

The Java interface declaration may look like this:

```
public interface Swimming {
    public void treadWater();
    public void swimFreestyle();
    public void breathe();
    ....
}
```

In the swimming simulation, dogs, humans, whales, trout and ducks implement the Swimming protocol or interface method declarations and would swim in ways unique to those species. A hummingbird, on the other hand, would probably not implement the Swimming protocol or interface, and therefore drown if left in water for a prolonged period of time.



3. If an accessor method isn't found and the class method `accessInstanceVariablesDirectly` returns true or YES, `takeValueForKey()` searches for an ivar based on the expected key name and sets the value directly in Java. In Objective C, the old value is autoreleased and the new one is retained. For the key "lastName", this would be `_lastName` or `lastName`.
4. If neither an accessor method or an ivar is found, the default implementation invokes `handleTakeValueForUnboundKey` in Java and `handleTakeValue:forUnboundKey:` in Objective C.

It is important that while this method has been very well debugged, sometimes phantom keys can be looked up long after they "died". It is prudent to override `handleTakeValueForUnboundKey` in this situation to check if the phantom key is being looked up and, if so, do nothing.

To set the ivar 'someDecimal' to a `BigDecimal` object of value '1', you would:

Java

```
myEO.takeValueForKey(new BigDecimal("1"), "someDecimal");
```

Valentina

object-relational database engine



The fastest database engine for the Mac OS

It operates 100's and sometimes a 1000 times faster than other systems

Valentina - scriptable DBMS..... \$49

- Includes special features for Web Developers.
- Glues for Frontier and MacPerl.

Valentina C++ SDK \$499/\$699

- Cross-platform (Mac OS/WIN32);
- bool, byte, short, long, float, double, date, time string, BLOB, TEXT, Picture.
- RegEx search, full text indexing, support international languages.
- SQL, random-access cursors.
- Multi-threading capable.
- Record locking.
- No runtime fees.

Valentina for REALbasic plugin \$199

Valentina for Macromedia Director Xtra \$199/299

Valentina for WebSiphon \$299

Valentina XCMD \$199

Make your application's database operations blazingly fast!

Order Directly
from Our Web Site

www.paradigmasoft.com
Hosted by MacServe.net

Download full featured evaluation version

Objective C

```
[myEO takeValue:[NSDecimalNumber  
decimalNumberWithString:@"1"] forKey:@"someDecimal"]
```

What happens when your code accesses ivars that no longer exist?

This is an interesting situation. Most often, this occurs when a developer has removed an ivar from his EO without deleting all references to it from the code. EOF's default behavior is to invoke `handleQueryWithUnboundKey` in Java and `handleQueryWithUnboundKey:` in Objective C which throws an exception. Developers can override this if they desire some other behavior or have it fail gracefully (not at all recommended but if it is *absolutely* necessary ...).

Corollary behavior occurs when using `takeValueForKey` (Objective C `takeValue:forKey:`) on an ivar that doesn't exist. EOF automatically invokes the method `handleTakeValueForUnboundKey` (Objective C `handleTakeValue:forUnboundKey:`). There are methods used as private API within EOF that are publicly exposed, `storedValueForKey` and `takeStoredValueForKey` in Java (`storedValueForKey:` and `takeStoredValue:forKey:` in Objective C) are used to access and/or initialize the receiver's values as they are stored in the database.

MAKING LIFE EASIER

There are some methods in `EOKeyValueCoding` that make life considerably easier to get things done. And in the case of Objective C, it provides a very convenient way to implement the `NSCoding` protocol within a class.

`valueForKeyPath` (Objective C `valueForKeyPath:`) allows you to return the value at the end of a relationship path's property. For example, if you wanted to find the name of an `Employee`'s department, you would:

Java

```
(String)anEmployee.valueForKeyPath("department.name");
```

Objective C

```
[anEmployee valueForKeyPath:@"department.name"];
```

The relationship from an `Employee` to its department was traversed and the department's name accessed. Having done this, you can use another method `takeValueForKeyPath` (Objective C `takeValue:forKeyPath:`) to set the name of a relationship's property. To change the name of an `Employee`'s department:

Java

```
anEmployee.takeValueForKeyPath("Data  
Processing", "department.name");
```

ObjectiveC

```
[anEmployee takeValue:@"Data Processing:  
forKeyPath:@"department.name"];
```

`valuesForKeys` (Objective C `valuesForKeys:`) returns an `NSDictionary` of the key-value pairs of an object. It's corollary

WHAT ARE IVARS/INSTANCE VARIABLES?

Instance variables (also known as ivars) are the specific variables of data for an object instance. What does that mean?

Let's say you have an `Employee` class which generically defines what an employee looks like. An `Employee` has a first name, a last name, and a Social Security number. In Objective C, an `Employee` would look like this:

```
@interface Employee : NSObject  
{  
    NSString *firstName;  
    NSString *lastName;  
    NSString *ssn;  
}  
.....  
@end
```

And in Java, an `Employee` would look like this:

```
public class Employee extends Object {  
    protected String firstName;  
    protected String lastName;  
    protected String ssn;  
    .....  
}
```

`Employee` objects, like the employees that represent the CEO and the vice-president, would have a `firstName`, `lastName`, and an `ssn` instance variable. For the CEO, her instance variables might look like this: `firstName = "Jane"`, `lastName = "Smith"`, `ssn = "555-12-1234"`. For the vice-president, his instance variables may look like this: `firstName = "John"`, `lastName = "Brown"`, `ssn = "555-12-9876"`. For the CEO `Employee` object, its variables contain unique data, and likewise for the vice-president `Employee` object as well.

`EOKeyValueCoding` provides a consistent API for the frameworks to access the ivars for the retrieval and modification of data. Whenever a `firstName` is displayed on a web page in a `WebObjects` app, it is retrieved through `EOKeyValueCoding`'s APIs. The corresponding modification of ivars occurs through `EOKeyValueCoding`'s APIs on all objects. This consistency allows developers to create applications and frameworks faster because they don't have to think about how the ivars need to be accessed.

`method takeValuesFromDictionary` (Objective C `takeValuesFromDictionary:`) sets the ivars of an object to the values of the `NSDictionary` passed as an argument.

These methods can be used to quickly implement the `NSCoding` protocol in Objective C. The `NSCoding` protocol can be used to make an object archivable to disk. It consists of two methods `encodeWithCoder:` and `initWithCoder:`. Imagining a situation where a developer wished to implement an `Employee` class that didn't use EOF. Some form of persistence needs to be created for the `Employee` objects by implementing the `NSCoder` protocol.

development revolution.

You're part of the movement.

At the core you'll find two innovative technologies allowing you to deliver the most comprehensive software solutions available – 4D and WebSTAR.

Time is short. Seize the moment.

Maximize your efforts using revolutionary development tools and Internet Servers that deliver now and into the future.

Don't waste today.

Download your future. Ignite your potential.



W E B S T A R . 4 D . C O M
1.800.881.3466

```
Employee.h
#import <Foundation/Foundation.h>
@interface Employee : NSObject <NSCoding>
{
    NSString *firstName;
    NSString *lastName;
    NSString *ssn;
    NSString *streetAddressOne;
    NSString *streetAddressTwo;
    NSString *city;
    NSString *state;
    NSString *zipCode;
    NSNumber *salary;
}
```

```
// EOKeyValueCoding-based methods
- (NSArray *)attributeNames
{
    // Return the ivar 'keys' in an NSArray
    return [NSArray arrayWithObjects:
        @"firstName",
        @"lastName",
        @"ssn"
    ];
}
```

```

        @"streetAddressOne",
        @"streetAddressTwo",
        @"city",
        @"state",
        @"zipCode",
        @"salary",
        nil];
}

- (NSDictionary *)objectAsDictionary
{
    // Return the object as an NSDictionary, using the -attributeNames
    // method to define the keys of the object
    return [self valueForKey:[self attributeNames]];
}

- (NSString *)description
{
    // Inherited from NSObject
    // Represent the object as an NSDictionary
    return [[self objectAsDictionary] description];
}

// NSCoder methods
- (void)encodeWithCoder:(NSCoder *)coder
{
    // NSDictionary already conforms to the NSCoder protocol, as
    // well as NSString and NSDecimalNumber
    [super encodeWithCoder:coder];
    [coder encodeObject:[self objectAsDictionary]];

    return;
}

- (id)initWithCoder:(NSCoder *)coder
{
    NSDictionary *newSelf;

    self = [super initWithCoder:coder];
    newSelf = [coder decodeObject];

    // Since we've already archived ourselves as an NSDictionary,
    // we can probably take our values from the decoded NSDictionary
    [self takeValuesFromDictionary:newSelf];

    return self;
}

@end
```

This is a simpler (and faster) way of implementing NSCoder, particularly as the number of ivars increases. In our example, the traditional approach would have required 10-12 additional lines of code for dealing with all of the ivars individually. Instead of all that, the object is archived as an NSDictionary and the NSCoder protocol takes care of the rest.

EOKeyValueCoding provides a consistent and convenient API to access and manipulate all of your ivars within your objects. It also provides a uniform way to represent all objects as NSDictionaries to WebObjects and EOF. This helps make object-oriented programming within WebObjects much simpler which is beneficial regardless of your level of expertise.

Unsurprisingly, EOKeyValueCoding is too useful to be kept to EOF and WebObjects alone. Apple recognized its potential by migrating it for JavaClient as NSKeyValueCoding. Even more significant, NSKeyValueCoding appears to be moving into Foundation for use with Cocoa programming. Other than some source code #import changes, all the API stays consistent which now means that persistence within classes will become much simpler.

Cross-Platform C++

*PP2MFC puts your
PowerPlant™
applications on
Windows®*

www.oofile.com.au

By Tom Djajadiningrat and Maarten Gribnau

Cubby: Multiscreen Desktop VR Part III

Reading an input sprocket device and calibrating Cubby

INTRODUCTION

In this month's final episode of our 'Cubby: Multiscreen Desktop VR' trilogy we explain how you read the InputSprocket driver from part II, how you use it as input for the cameras from part I and how you calibrate the input device so that it leads to the correct head position.

RELATING THE VIRTUAL CUBBY TO THE REAL WORLD

Before we can talk calibration we need to establish how the virtual Cubby relates to the real world. We have made life easy for ourselves by choosing the same orientation for the coordinate system of the real Cubby as the virtual Cubby. We also made the dimensions of the virtual Cubby in QuickDraw3D units equal to the dimensions of the real world Cubby in millimetres. This is determined by the constants `kHalfEdgeLength` and `kEdgeLength` which are half an edge length and a whole edge length of Cubby respectively. You can find these constants in `MyDefines.h` (Listing 1). The Cubby we built has an edge

length of 195mm and so we've given our virtual Cubby an edge length of 195 QuickDraw3D units. In a sense the scale of the virtual Cubby is arbitrary. As long as everything is scaled equally (the background planes, the model, the lights and the camera) you end up with the same perspectives. If you like you can create a virtual Cubby with an edge length of 1. However, we think that the way we do it here has one major advantage: the coordinates in QuickDraw 3D units that you see during debugging are meaningful because you can directly relate them to sizes in millimetres in the real world. For example, if you end up with a camera position that is 20000 QuickDraw 3D units from the origin, you know that something has gone haywire because 20 metres is well out of range of the tracker.

Listing 1: MyDefines.h

```
// the width and half the width of a Cubby edge in QD3D units
#define kEdgeLength 195
#define kHalfEdgeLength 195/2.0
```

We use these constants for three things:

1. setting the size of the background planes
2. scaling the model read from disk to Cubby's display space
3. setting the area cut out of the view plane

Setting the size of the background planes

Listing 2 shows the creation of the background planes. Each background plane is a polygon. The constant `kEdgeLength` is used to indicate the vertices of a polygon.

Listing 2: DisplaySpace.c

```
DisplaySpace
TQ3GeometryObject thePolygonZ0, thePolygonX0, thePolygonY0;
TQ3PolygonData theData;
long i;

TQ3Vertex3D theVerticesZ0[4] = {
    0, 0, 0, nil,
    kEdgeLength, 0, 0, nil,
```

Last month, when **Maarten** threatened to withdraw his indispensable contribution to this month's final episode of Cubby, **Tom** hurried to explain to him that the trick in crafting 'about the author' notes is to spend at least as much time on it as on the article itself. He is disappointed that his suggestion for an article with guidelines for writing 'about the author' notes was received with little enthusiasm by MacTech's editors.

The products you need, with the prices and service you deserve... guaranteed.

Power Tools for Programmers!

CodeWarrior Pro 6

NEW!

CodeWarrior for Mac OS is a powerful integrated development environment that includes a fully object-oriented application development framework called PowerPlant. With CodeWarrior for Mac OS you can quickly develop reliable, professional quality applications that execute on Classic Mac OS, or OS X.

NEW Version



ONLY!

\$359

REALbasic 2.1

NEW!

REALbasic is the programming tool for "the rest of us." Each window type, control, and menu is preconfigured and instantly works as it should. The drag and drop Window Editor allows you to quickly and easily create your application's interface so you can focus on the important part—your creativity. Includes 900 page manual, examples and tutorial on CD ROM! The professional version has all the power of the standard, plus database support and allows you to cross compile your code for Windows with a single click!

As low as



\$139

Spotlight

Spotlight is the first Macintosh "Automatic Debugger". It can automatically locate run time errors in your code and display the offending source code line. Unlike similar tools on other platforms Spotlight is easy to use. No source code changes are necessary for application debugging. Spotlight can automatically check for wild pointers, memory leaks, overwrites, underwrites, invalid dereferencing of handles, and even toolbox parameter validity checking – spotlight knows Macintosh verifying parameters to over 400 toolbox calls.



\$189

VOODOO Server

VOODOO Server is a version control system for software developers using Metrowerks CodeWarrior under Mac OS. VOODOO Server and the corresponding VOODOO clients (the included CodeWarrior VCS plug-in and the VOODOO Admin application) are designed to offer reliable and robust version control features while minimizing the administrative overhead that usually accompanies version control. If you're a single programmer or managing a team of developers, version control can make or break your project. Do it right, with VOODOO



\$79

Resorcerer 2.2

Resorcerer is the only supported general-purpose resource editor for Macintosh. Relied upon by thousands of Mac developers, Resorcerer features a wealth of powerful yet easy-to-use tools for easier, faster, and safer editing of Macintosh data files and resources. Whether you have to parse a picture, debug a data fork, design and try out Balloon Help, create a scripting dictionary, create anti-aliased icons, design and edit a custom resource with 40,000 fields in it, create C source code to run a dialog, or any of hundreds of other resource-related tasks, Resorcerer's magic will quickly save you time and money.



\$239

Future BASIC 3

NEW!

One of the most flexible and powerful development environments on the Macintosh today! Easily create programs with the visual program editor, drop into the BASIC editor to define powerful logic with the worlds easiest programming language, or work directly with the Macintosh toolbox. The only BASIC compiler on the market that gives you 100% access to all of the power of the Macintosh toolbox!



\$159

and hundreds more!



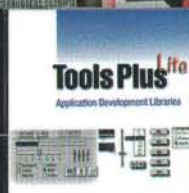
Page Charmer 2.0
\$139



Scripter 2.0
\$179



Script Debugger
\$189



ToolsPlus Lite
\$99



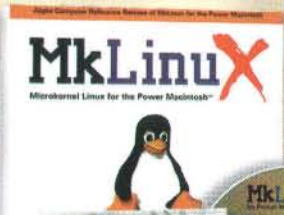
FaceSpan 3.0
\$179



WebSpice 1,000,000
\$89



WebSpice Animations
\$89



MkLinux
\$39



PowerKey Rebound
\$89

Developer DEPOT®

PO Box 5200 • Westlake Village, CA • 91359-5200 • Voice: 800/MACDEV-1 (800/622-3381)
Outside US/Canada: 805/494-9797 • Fax: 805/494-9798 • E-mail: orders@devdepot.com

www.devdepot.com

Master the Web!

WebSTAR Server Suite 4.3

WebSTAR Server Suite is a complete set of powerful and easy-to-use Internet servers for the Mac OS. Effortlessly serve web pages, host email accounts, publish databases, and share files - all with a single application on one Mac! WebSTAR Server Suite is perfect for Internet or Intranet serving, single or multiple sites, small and large businesses - it's power and ease-of-use saves any organization time and money.



\$539

CyberGauge 3.0

Monitor the bandwidth usage of up to five different machines on your network! Do you need to upgrade your webserver? How hard is your eMail server working? Are you getting all the bandwidth you're paying for? Not only can CyberGauge answer all these questions, new features allow CyberGauge to eMail or page your network device becomes unresponsive or passes a threshold of usage you define - an essential first line of defense for early detection of denial of service attacks and necessity for warning you and tracking quality of ISPs that may have brown outs and shutdowns.



\$249

Funnel Web Pro

Funnel Web is the ultimate web analysis solution for professionals. Specifically designed for profiling web site usage and monitoring customer usage patterns, Funnel Web is ideal for examining server performance and online effectiveness. Funnel Web can analyze log file formats from any server, WebSTAR, WebTen, even Unix or NT hosted servers. Discover the most popular pages on your site, track server loads & optimize server performance, profile visitors based on organization, domain name, country, browser, etc. Funnel Web does it all!



As low as

\$329

NetBarrier

NetBarrier offers a Personal Firewall, Antivandal protection, and Internet Filtering. Protect your machine from intrusions by Internet or AppleTalk. Incorrect passwords and individual actions are logged, you are alerted to hostile actions, and intruders are easily isolated. Internet Filtering allows you to be sure that passwords, credit card numbers, and other sensitive information can never be exported from your computer - the content itself is filtered before any transfer! (Rated 4 mice by Macworld Magazine)



\$49⁹⁵

...and great hardware solutions!

2 USB PCI Card

Add two USB ports to your older Macintosh. Connect up to 127 devices to the Universal Serial Bus (USB) that is Apple's new standard for desktop connectivity. USB mouse devices, keyboards, joysticks, game controllers, printers, scanners - connect them all to your current computer. Installs in minutes!



\$32⁹⁵

Macsense USB Full Size Keyboard

Just plug this keyboard into your Mac and start typing! The UKB-600 keyboard from Macsense is designed to get you typing quickly and easily, without any hassle or compatibility worries. It features two tone translucent design, colored to match your favorite flavor of Macintosh. It offers soft touch with positive tactile feedback and build built in USB port on either side of the keyboard. Includes a 5' USB cable and is 100% Macintosh compatible, simply plug and play, as easy as Macintosh!



\$44⁹⁵

Dr. Bott Moni Switch ADB or USB

Do you need 4 monitors and 4 keyboards for your 4 servers? With Dr. Bott Moni-Switch you can connect a single keyboard and monitor to up to 4 machines at once! A simple flick of a switch directs the video input and keyboard commands to the appropriate CPU! Available in USB and ADB models, with 2 or 4 machine support, and bundles with USB PCI cards so you can mix and match USB and ADB machines with the same Moni-Switch! Great for programmers to do back ground compiles, ideal for server rooms overcrowded with monitors and keyboards!

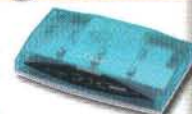


As low as

\$129⁹⁵

Macsense Internet Sharing Router

Looking to get your whole office online without shelling out thousands of dollars? If so, the XRouter Internet Sharing Hub offers the perfect solution. This amazing Ethernet-to-Ethernet hub connects an entire network of up to 252 users to the Internet using only one ISP account and one Cable or DSL modem!



\$199

```

    kEdgeLength, kEdgeLength, 0, nil,
    0, kEdgeLength, 0, nil);

TQ3Vertex3D theVerticesX0[4] = {
    0, 0, kEdgeLength, nil,
    0, 0, 0, nil,
    0, kEdgeLength, 0, nil,
    0, kEdgeLength, kEdgeLength, nil};

TQ3Vertex3D theVerticesY0[4] = {
    0, 0, kEdgeLength, nil,
    kEdgeLength, 0, kEdgeLength, nil,
    kEdgeLength, 0, 0, nil,
    0, 0, 0, nil};

// create new polygon objects.

// four corners per polygon.
theData.numVertices = 4;

// point to our array of vertices
// for the polygon in the Z=0 plane.
theData.vertices = theVerticesZ0;

// polygon itself has no attributes.
theData.polygonAttributeSet = nil;

// create the polygon.
thePolygonZ0 = Q3Polygon_New(&theData);

// point to our array of vertices
// for the polygon in the X=0 plane.
theData.vertices = theVerticesX0;

// create the polygon.
thePolygonX0 = Q3Polygon_New(&theData);

// point to our array of vertices
// for the polygon in the Y=0 plane.
theData.vertices = theVerticesY0;

// create the polygon.
thePolygonY0 = Q3Polygon_New(&theData);

```

Scaling the model

We want to scale and translate the model that was read from disk so that it fits within and is centred within Cubby's display space. This is accomplished through the procedure `ScaleModelToDisplaySpace` within the source file `ReadModelAndScaleIt.c` (Listing 3). It should look pretty familiar to you as similar procedures are found in most examples from the QuickDraw 3D SDK. Basically, what we do here is calculate the bounding box of the model and use the dimensions of the bounding box to scale and translate the model so that it fits and is centred within Cubby's display space. Let's look at the code in detail.

The first thing we do is call `GetModelBoundingBox` after which the `ViewBBox` holds the bounding box. From the bounding box we can work out its dimensions along the three world axes. The next thing is a check to see whether all these dimensions are smaller or equal than `kQ3RealZero`. This could happen if the file read from disk was a single point. We want to avoid a bounding box of which all three dimensions are smaller than zero as that would give scaling problems later. So if it happens we set the dimensions of the bounding box to a very small number (0.0001). We work out the bounding box' centre the `BBoxCenter` by calling `Q3Point3D_AffineComb` with the

box' minimum and maximum corners. We also calculate the `BBoxDiagonal`, the length of the diagonal of the bounding box. We then calculate the `DisplaySpaceDiagonal`, the length of the diagonal of Cubby's display space. The ratio of these two lengths gives us the scale factor for the model so that its bounding box fits within Cubby's display space. Note how we have a fiddle factor `kScaleFineTune`. This factor makes sure that the model ends up slightly smaller than Cubby's display space as it does not look very good when the model touches the background planes. We chose `kScaleFineTune = 0.75` though you can of course change it should you prefer a tighter or looser fit. We can now work out the required transformation matrices. The first one, the `TransMatrix1`, translates the centre of the model to the origin. The second one, the `ScaleMatrix`, scales the model around the origin. The third matrix, the `TransMatrix2`, translates the model to the centre of the display space. Finally, we concatenate these three matrices in the order that they were created. The resulting matrix in the `fModelMatrix` field of `gDoc` is submitted for rendering in `SubmitOneView` of the source file `Rendering.c`.

Listing 3: ReadModelAndScaleIt.c

```

ScaleModelToDisplaySpace
void ScaleModelToDisplaySpace(DocumentPtr inDoc)
{
    #define kScaleFineTune 0.75

    TQ3BoundingBox theViewBBox;
    float theXSize, theYSize, theZSize;

    float theWeights[2] = { 0.5, 0.5 };
    TQ3Point3D thePoints[2];
    TQ3Point3D theBBoxCenter;
    TQ3Vector3D theDiagonalVector;
    float theBBoxDiagonal;
    float theDisplaySpaceDiagonal;

    float theScale;

    TQ3Matrix4x4 theTransMatrix1,
    theTransMatrix2,
    theScaleMatrix;

    // Get the bounding box of the model.
    GetModelBoundingBox(inDoc, &theViewBBox);

    // Work out the dimensions along the axes.
    theXSize = theViewBBox.max.x - theViewBBox.min.x;
    theYSize = theViewBBox.max.y - theViewBBox.min.y;
    theZSize = theViewBBox.max.z - theViewBBox.min.z;

    // If we have a point model, then the 'theViewBBox'
    // would end up being a 'singularity' at the location
    // of the point. As this bounding 'box' is used in
    // scaling the model, this could give problems.
    if (theXSize <= kQ3RealZero &&
        theYSize <= kQ3RealZero &&
        theZSize <= kQ3RealZero)
    {
        // Set the corners of the bounding box to a very small number.
        theViewBBox.max.x += 0.0001;
        theViewBBox.max.y += 0.0001;
        theViewBBox.max.z += 0.0001;

        theViewBBox.min.x -= 0.0001;
        theViewBBox.min.y -= 0.0001;
        theViewBBox.min.z -= 0.0001;
    }
}

```

```

//Work out the centre of the bounding box.
thePoints[0] = theViewBBBox.min;
thePoints[1] = theViewBBBox.max;

Q3Point3D_AffineComb( thePoints, theWeights,
    2, &theBBBoxCenter);

// the length of the diagonal of the bounding box.
Q3Point3D_Subtract( &theViewBBBox.max,
    &theViewBBBox.min,
    &theDiagonalVector);
theBBBoxDiagonal= Q3Vector3D_Length(&theDiagonalVector);

// the length of the diagonal of the display space.
theDisplaySpaceDiagonal = sqrt(3 * pow(kEdgeLength,2));

//The scale to make the model fit within the space.
// kScaleFineTune = 1 gives a tight fit. The smaller you
// make it, the smaller the model ends up
// within the display space.
theScale = kScaleFineTune * (theDisplaySpaceDiagonal/
    theBBBoxDiagonal);

// First, we set up a matrix for translating
// the centre of the model to the origin.
Q3Matrix4x4_SetTranslate ( &theTransMatrix1,
    -theBBBoxCenter.x,
    -theBBBoxCenter.y,
    -theBBBoxCenter.z);

// Second we set up a matrix for scaling
// the model to fit Cubby's space.
Q3Matrix4x4_SetScale ( &theScaleMatrix,
    theScale,
    theScale,
    theScale);

//Third, we set up a matrix for translating the model
// to the middle of Cubby's display space.
Q3Matrix4x4_SetTranslate ( &theTransMatrix2,
    kEdgeLength/2.0,
    kEdgeLength/2.0,
    kEdgeLength/2.0);

// Finally, concatenate the matrices.
Q3Matrix4x4_Multiply( &theTransMatrix1,
    &theScaleMatrix,
    &inDoc->fModelMatrix);

Q3Matrix4x4_Multiply( &inDoc->fModelMatrix,
    &theTransMatrix2,
    &inDoc->fModelMatrix);
}

```

Setting the view plane area

We need to indicate the size of the area that each view plane camera cuts out of its view plane. We need to do this only once during initialization of each camera since the size of this area does not change during execution of our application. The size of the area that is cut out of the view plane is determined by the camera parameters `halfWidthAtViewPlane` and `halfHeightAtViewPlane`. Since the area we wish to cut out is square and since a background polygon completely fills it, these camera parameters both equal `kHalfEdgeLength` (Listing 4).

Listing 4: ViewCreation.c

```

newViewPlaneCamera

//The aspect ratio of these parameters should equal
// that of the paneWidth and paneHeight.
// In Cubby's case the screens are square so the aspect ratio = 1.
thePerspectiveData.halfWidthAtViewPlane = kHalfEdgeLength;
thePerspectiveData.halfHeightAtViewPlane = kHalfEdgeLength;

```

Now that we know how the virtual Cubby relates to the real world, we can turn our attention to calibration.

CALIBRATION PROCEDURE

Why calibrate?

To ascertain that we correctly measure the head-position of the user we need to calibrate the head-tracker. Calibration would not be necessary if we could place the origin of the base-unit of the tracker at the origin of Cubby and the sensor in the middle of the user's eye. Since neither is possible we need to take the offsets in position into account. Also, since we wish to mount the Dynasight base unit above the display space under an angle to provide the best coverage of the user's work space, we need to take this rotation into account. What we are looking for is the matrix which transforms the InputSprocket coordinates that we get from the tracker to the QuickDraw 3D world coordinate system. So how do we work out this transformation matrix?

Conversion from InputSprocket units to millimetres

The numbers that we read from our Dynasight InputSprocket driver into our application are not in millimetres. The driver works with a coordinate system in which the axes run from the minimum number

Engineer for Hire

Henry Carstens
Contract Software Engineer

Cross-platform C, C++
CodeWarrior, PowerPlant
Classic, Carbon, OSX

13 Years Experience
20 Commercial Software Apps Shipped
Fast, Dependable, Self-Motivating

On Site or Off

Henry Carstens
carstens@verticalsolutions.com
503-524-6831

Because how soon you ship *does* matter.

kISpAxisMinimum to the maximum number kISpAxisMaximum. A look in InputSprocket.h tells us:

```
#define kISpAxisMinimum 0x00000000U
#define kISpAxisMaximum 0xFFFFFFFFU
```

Unlike most game applications, we are not interested in relative movements but in absolute positions. In order to calculate what the coordinates from the InputSprocket driver correspond to in the real world we need to agree upon the conversion factor. What we have done is to set kISpAxisMaximum to 10 metres. Though in a sense our choice is arbitrary we would like to think this is an informed choice for two reasons. First, it is well above the maximum range of the Dynasight (6m) even with the largest possible reflective target (75 mm diameter). Second, it gives us plenty of resolution. $kISpAxisMaximum / 10 \text{ metres} = 4,294,967,295 / 10,000 \text{ millimetres} = 429496.7295 \text{ steps/mm}$. Tracking technology would have to improve pretty dramatically for us to run out of resolution.

So if:

$4,294,967,295 \text{ InputSprocket units} = 10 \text{ metres}$

then

$1 \text{ InputSprocket unit} = 10,000 \text{ mm} / 4,294,967,295 = 2.328306437080797\text{e-}6 \text{ mm}$

This is our multiplication factor to convert from InputSprocket units to millimetres.

Working out the transformation matrix

The first thing we do is to convert the Dynasight InputSprocket coordinates to millimetres. For this we create a matrix theScaleMatrix.

We then need to express the Cubby coordinate system in Dynasight InputSprocket coordinates. The easiest way would be to measure points on Cubby's main axes, but since these are out of sight of the Dynasight we have to use points on the top edges of Cubby's screens. We use the points P, Px and Pz (**Figure 1**). In the real-world coordinate system in millimetres $P = \{0, dy, 0\}$, $Px = \{195, dy, 0\}$ and $Pz = \{0, dy, 195\}$, where dy is the distance in millimetres above Cubby's ground plane. In our case dy is 204mm, though your setup will of course be different. Though we are working on integrating a perky animated paper clip within the Cubby application which will talk you through the calibration procedure, for the moment we take the following approach. We place the sensor consecutively at P, Px and Pz, measure the Dynasight InputSprocket coordinates of these points by means of our separate little app DynasightReader and jot down the readings. We then fill in these coordinates where the variables P, Px and Pz are initialized in CalibrationMatrix (Listing 5). For a real app you would of course have to think of a more elegant way to capture the

Dynasight InputSprocket coordinates.

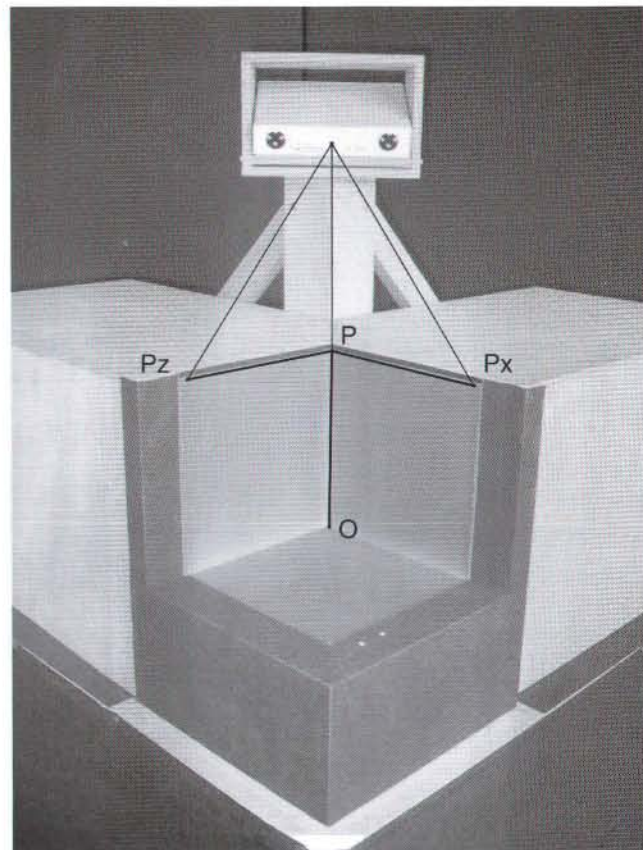


Figure 1. The points to work out the calibration matrix.

Using these three points we then calculate the vectors PPx and PPz and normalize them into Xnorm and Znorm respectively. Using the cross-product of these vectors, we calculate a unit vector Ynorm which is perpendicular to both Xnorm and Znorm. Now we have an orthonormal base formed by the unit vectors Xnorm, Ynorm and Znorm which specifies the orientation of Cubby. The rotation matrix to rotate from orientation of the Dynasight InputSprocket coordinate system into the orientation of the Cubby coordinate system can be constructed by taking these unit vectors and using them as the columns of the rotation matrix. To save some typing we first set the rotation matrix to a 4x4 identity matrix and then substitute the top left 3x3 matrix with the orthonormal base.

Now work out the vector PO from P to the origin of Cubby expressed in the Dynasight InputSprocket coordinate system by multiplying the unit vector in the y-direction Ynorm by the distance from P to the origin in Dynasight coordinates. By adding the vector PO to point P we find the origin expressed in the Dynasight InputSprocket coordinate system. This is of course a very informative point but we could not measure it directly since there is no direct line of sight to the Dynasight.

**AS A
PROGRAMMER
OR SYSADMIN,
YOU'VE GOT
BETTER THINGS
TO DO THAN
FIGHT WITH A
WEB SITE**



It's time for you to take a look at MGI

MGI, a plug-in to 4D's award-winning server, WebSTAR, is designed to provide functionality to otherwise static web sites, whether for a private intranet or enterprise - class ISP/ASP. MGI was specifically developed to be used by web graphic designers with no scripting or programming experience. If you know HTML, you know MGI. And if you are a programmer, you can learn MGI by the time your pizza is delivered. You can try out MGI for free - right now - without obligation by downloading a fully - functioning demo at :

<http://www.pageplanetsoftware.com>



SOFTWARE BUILT WITH YOU IN MIND

Searchable Databases
Online Stores
Info Baskets
Guestbooks
Banner Ads
Credit Card Transactions
Counters
Password Protection
Surveys
Quizzes
Form Processing
Conditionals
Math
Cookies

Date and Time
File Includes
File Writes
Web Based Email
Classified Ads
Discussion Groups
Web Chat
Affiliate Tracking
I/O Transactions
Δ Time Calculations
PGP Encryption
Token Tracking
Calendars
Random Rotation
Send Mail
E - Cards
Credit Bank

What we are ultimately looking for is a calibration matrix which when applied to the origin measured in raw Dynasight units gives us {0,0,0}. So what we do is to rotate the origin using the rotation matrix theRotMatrix we just created and then use the resulting point temp to create a translation matrix theTransMatrix which translates the point temp to the origin of the Dynasight coordinate system.

Finally, we concatenate the three matrices. First we scale, then we rotate, and finally we translate. The resulting matrix theCalMatrix is returned.

Listing 5: Calibration.c

```

CalibrationMatrix
TQ3Matrix4x4 CalibrationMatrix()
{
    //These coordinates in millimetres were measured with the DynasightReader app.
    TQ3Point3D P = {-0.7, -144.6, 218.15};
    TQ3Point3D Px= {136.2, -94.75, 351.55};
    TQ3Point3D Pz= {-138.45, -93.45, 343.75};

    TQ3Matrix4x4 theTransMatrix, theRotMatrix, theCalMatrix;

    //Vertical distance from P to ground plane in mm.
    float dy = 204;

    TQ3Vector3D PPx, PPz, Xnorm, Ynorm, Znorm, PO;
    TQ3Point3D temp, O;

    // parallel to X, 195mm
    Q3Point3D_Subtract(&Px, &P, &PPx);

    // X normalized
    Q3Vector3D_Normalize(&PPx, &Xnorm);

    // parallel to Z, 195mm
    Q3Point3D_Subtract(&Pz, &P, &PPz);

    // Z normalized
    Q3Vector3D_Normalize(&PPz, &Znorm);

    //Y normalized
    Q3Vector3D_Cross(&Znorm, &Xnorm, &Ynorm);

    // Start with setting the rotation matrix to identity
    // so that we do not have to fill in all 16 fields.
    Q3Matrix4x4_SetIdentity(&theRotMatrix);

    //The columns of the rotation matrix become
    // the vectors of the orthonormal base of the Cubby
    // coordinate system in the Dynasight coordinate system.
    theRotMatrix.value[0][0] = Xnorm.x;
    theRotMatrix.value[0][1] = Ynorm.x;
    theRotMatrix.value[0][2] = Znorm.x;
    theRotMatrix.value[1][0] = Xnorm.y;
    theRotMatrix.value[1][1] = Ynorm.y;
    theRotMatrix.value[1][2] = Znorm.y;
    theRotMatrix.value[2][0] = Xnorm.z;
    theRotMatrix.value[2][1] = Ynorm.z;
    theRotMatrix.value[2][2] = Znorm.z;

    //Work out the vector PO (from P to the origin of Cubby)
    // expressed in the Dynasight coordinate system.
    // For this you need the distance dy from P to the
    // origin of Cubby in mm.
    Q3Vector3D_Scale(&Ynorm, -dy, &PO);

    // Now we can work out the origin of Cubby
    // expressed in the Dynasight coordinate system.
    Q3Point3D_Vector3D_Add(&P, &PO, &O);

    // Rotate the origin into the Cubby coordinate system...
    Q3Point3D_Transform(&O, &theRotMatrix, &temp);

```

```

    // and use the resulting point to create the translation matrix.
    Q3Matrix4x4_SetTranslate(&theTransMatrix,
        -temp.x,
        -temp.y,
        -temp.z);

```

```

    // Concatenate the three matrices:
    // first we rotate, then we translate.
    Q3Matrix4x4_Multiply(&theRotMatrix,
        &theTransMatrix,
        &theCalMatrix);

```

```

    return theCalMatrix;
}

```

INTEGRATION OF INPUTSProCKET CODE WITH CUBBY

In the previous episode (Gribnau and Djajadiningrat, 2000), we introduced the basics of communication between InputSprocket applications and drivers. Now, we will show how Cubby configures and reads data from InputSprocket drivers. For Cubby, it doesn't matter if the data comes from our Dynasight driver or from any other InputSprocket driver.

Initializing InputSprocket

Cubby, like any other InputSprocket application, needs to initialize InputSprocket before. Listing 6 shows the ISp_Initialize routine that does the initialization within the Cubby application. First of all, InputSprocket is loaded with a call to ISpStartup. Then, Cubby asks InputSprocket to create new virtual **elements** based on its input **needs** with the ISpElement_NewVirtualFromNeeds call. As shown in our previous episode, InputSprocket defines every input device in terms of its elements (e.g. buttons, axes, directional pads, etc.). An application can ask InputSprocket to create new elements based on its needs. With the needs and the elements, Cubby can call ISpInit. In response, InputSprocket will initialize the drivers and the drivers will auto-configure to Cubby's needs. This means that the drivers will try to find an optimal match between the needs of Cubby and the elements that they have.

Listing 6: ISpCubby.c

```

ISp_Initialize
OSStatus ISp_Initialize(
    OSType creator)
{
    OSStatus err;

    err = ISpStartup();
    if (err) {
        return err;
    }

    // Setup the input sprocket elements
    err = ISpElement_NewVirtualFromNeeds(
        kNeed_NeedCount,
        gNeeds, gInputElements, 0);
    if (err) {
        return err;
    }

    // Initialize InputSprocket and give it our needs
    err = ISpInit(
        kNeed_NeedCount, // number of needs

```

```

gNeeds,           // array of needs
gInputElements,  // array of references to virtals
creator,         // creator code
'0015',         // subcreator code
0,              // flags (should be 0)
kResourceID_set1, // a set list resource
0);             // reserved
if (err) {
    return err;
}

// We leave keyboard and mouse disabled for ISp
return noErr;
}

```

Listing 7 shows the needs of Cubby. Actually, it is an array of three structures of type `ISpNeed`. There are three needs since Cubby needs to know the position of the head of the user in three dimensions. Therefore, the need array has a need entry for every coordinate of the head position. Each need structure field is filled with appropriate values. The need for the x coordinate for instance, sets the name field to "Head X". Another important field is the icon field. This is set to the resource id of the icon that we want to be shown in the configuration dialog box. Furthermore, the element type (`kISpElementKind_Axis`) and the element label (`kISpElementLabel_Axis_XAxis`) are set.

Listing 7: ISpCubby.c

Cubby's need array

```

static ISpNeed gNeeds[kNeed_NeedCount] =
{
    ( "\pHead X", kResourceID_needIconBase+kNeed_HeadX, 0,
      0, kISpElementKind_Axis, kISpElementLabel_Axis_XAxis,
      0, 0, 0, 0 ),
    ( "\pHead Y", kResourceID_needIconBase+kNeed_HeadY, 0,
      0, kISpElementKind_Axis, kISpElementLabel_Axis_YAxis,
      0, 0, 0, 0 ),
    ( "\pHead Z", kResourceID_needIconBase+kNeed_HeadZ, 0,
      0, kISpElementKind_Axis, kISpElementLabel_Axis_ZAxis,
      0, 0, 0, 0 );
};

```

Configuration

The drivers might do a good job during auto-configuration. However, the major advantage of InputSprocket is that it allows users to reconfigure the mapping of application needs to driver elements. Listing 8 shows that one call suffices to start the configuration process. The `ISpConfigure` call brings the configuration dialog box on the screen if the Dynasight device (or any other device with an InputSprocket driver) is hooked up (**Figure 2**). Internally, InputSprocket starts the configuration process and negotiates with the driver as was explained in last month's episode. Cubby is not involved in this process which is nice because it does not have to know which device element is delivering data for each coordinate. The pull-down menus in the configuration dialog box allow users to select the elements that need to be mapped to a coordinate. If no InputSprocket device is available, InputSprocket will notify the user with a different dialog box (shown in **Figure 3**).

Listing 8: ISpCubby.c

ISp_ShowConfigureDialog

```

OSSStatus ISp_ShowConfigureDialog(void)
{
    return ISpConfigure(nil);
}

```

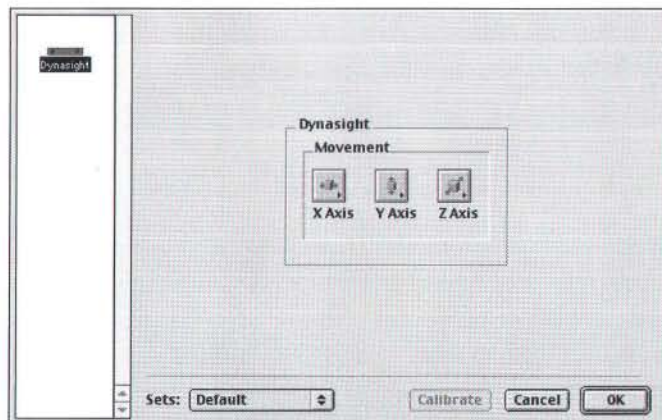


Figure 2. The configuration dialog box with the Dynasight selected.

StoneTable

You thought it was **just** a replacement for the List Manager ?

We lied, it is **much** more !

Tired of always adding just one more feature to your LDEF or table code ? What do you need in your table ?

Pictures and Icons and Checkboxes ?
 adjustable columns or rows ?
 Titles for columns or rows ?
 In-line editing of cell text ?
 More than 32K of data ?
 Color and styles ?
 Sorting ?
 More ??

How much longer does the list need to be to make it worth
 \$200 of your time ?

See just how long the list is for StoneTable.

Make StoneTable part of your toolbox today !

Only \$200.00

MasterCard & Visa accepted.

More Info & demo
<http://www.teleport.com/~stack>

StoneTable Publishing
 Voice/FAX (503) 287-3424
stack@teleport.com

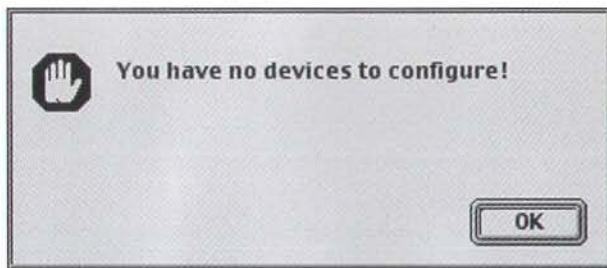


Figure 3. The dialog box shown if there are no InputSprocket devices.

Reading data from InputSprocket drivers

After initialization and configuration, Cubby can start to read data from the elements. Listing 9 shows how Cubby reads the x-coordinate of the head position. Reading the other coordinates follows the same procedure. First we check whether there are events waiting for the axis element with a call to `ISpElement_GetNextEvent`. In other words: we check whether the x-coordinate has changed since the last time this routine was called. If there was no error and an event was found, the raw axis value in InputSprocket coordinates is extracted with the `ISpElement_GetSimpleState` call. As was mentioned above, the raw coordinate needs to be converted to millimeters to be useful in Cubby. This is done by calling `ISp_AxisToCubbyAxis`. Finally, the events for the axis element are flushed and the new coordinate is stored.

Listing 9: ISpCubby.c

```
ISp_GetHeadX
void ISp_GetHeadX(ISpCubbyState *cubbyState)
{
    OSStatus      error = noErr;
    ISpElementEvent event;
    Boolean        wasEvent;
    ISpAxisData    axisValue;
    float          xValue = cubbyState->headX;

    // We check the axis, to see if it was moved, if so, we use that value
    error = ISpElement_GetNextEvent(
        gInputElements[kNeed_HeadX], sizeof(event),
        &event, &wasEvent);
    if (!error && wasEvent) {
        error = ISpElement_GetSimpleState(
            gInputElements[kNeed_HeadX], &axisValue);
        if (!error) {
            xValue = ISp_AxisToCubbyAxis(axisValue);
        }
        ISpElement_Flush(gInputElements[kNeed_HeadX]);
    }

    cubbyState->headX = xValue;
}
```

Listing 10 shows the conversion from raw coordinates to coordinates in millimeters. The raw coordinates are assumed to be in the format described above where `kISpAxisMaximum` equals 10 meters. Our `ISp_AxisToCubbyAxis` routine converts them to millimeters in three steps. First they are scaled to a value between 0 and 1. Then a midpoint is introduced. The InputSprocket coordinates are

positive only but axis values usually have a positive as well as negative values. Therefore, coordinates are transformed to fit between -1 and 1. Finally, a multiplication by 10,000 will give us the desired millimeters (assuming the driver follows the same convention). Now the coordinates are ready to be used to drive Cubby's cameras.

Listing 10: ISpCubby.c

ISp_AxisToCubbyAxis

```
float ISp_AxisToCubbyAxis(
    ISpAxisData axis)
{
    float value;
    static float r = kISpAxisMaximum - kISpAxisMinimum;

    value = (axis - kISpAxisMinimum) / r; // value between 0 and 1
    value *= 2;                          // value between 0 and 2
    value -= 1;                          // value between -1 and 1
    value *= 10000;                       // value is now between -10000 and 10000

    return value;
}
```

APPLYING THE CALIBRATION MATRIX

Now that we can read the raw coordinates from the Dynasight and have the calibration matrix, all that remains is to transform the raw coordinates into a calibrated head-position by applying the calibration matrix. This happens in `AdjustCameras` of `ViewPlaneCamera.c` (Listing 6). Using the `QuickDraw 3D` call `Q3Point3D_Transform` we apply the calibration matrix `fCalMatrix` to the head position `H` in raw coordinates and end up with a calibrated head position `C` in the Cubby coordinate system.

Listing 11: ViewPlaneCamera.c

AdjustCameras

```
// apply the calibration matrix to convert the
// raw head position to a camera position
// in world coordinates.
Q3Point3D_Transform( &H,
    &inDoc->fCalMatrix,
    &C);
```

TROUBLESHOOTING

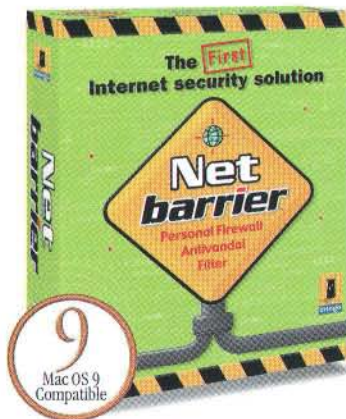
If things are behaving erratically make sure you have checked all the troubleshooting hints in the previous episodes. Here is one more possible pitfall.

If you've actually got round to building a Cubby setup you may find that lining up the three projections drives you completely up the wall. Well, it is indeed awkward but there is one thing which can make it unnecessarily difficult. Even though the `QuickDraw3D` panes may be square (228x228 pixels) this does not necessarily mean that the projected images are square. Somewhere in the chain of scan converter and projector you may lose some 'squareness'. If the projections of the panes are not square,

You think the Internet is safe. Think again...



NetBarrier. The first Internet security solution for Macintosh.



All Macs connected to the Internet (dialup, DSL, cable-modem) are exposed to hackers. Whether you are a home user or a professional user, your data interests them. That's why you need a security solution that only NetBarrier can provide.

Personal Firewall

NetBarrier protects and monitors all incoming and outgoing data. A customized mode allows you to create your own defense rules, thereby offering the most secure level of protection.

Antivandal

NetBarrier blocks all attempts to break into your Mac, detects wrong passwords and logs vandal attacks for complete protection. Moreover, it has an alarm to inform you of every intrusion attempt.

Internet filter

NetBarrier analyzes data as it leaves your computer and prevents unauthorized exporting of private information such as credit card numbers, passwords, sensitive data and more...

Available @ **Developer DEPOT**

<<http://www.devdepot.com>>
Toll Free: 877-DEPOT-NOW
Outside U.S./Canada: 805/494-9797

www.intego.com



you will never be able to make them line up. If this is the case, have a look whether your scan converter or projector has a control for the aspect ratio of the image. Most scan converters and projectors do. If not, you can always change the aspect ratio of the QuickDraw 3D panes in the code. The easiest thing to do here is to point the projectors at a wall and fiddle with the settings until the resulting aspect ratio is exactly 1:1. To lend you a hand with lining up the projections, we provide an optional 'lining up' texture with edge markings and centre lines (Figure 4). You can activate it in MyDefines.h by setting kCubePlanesTexture to 500. Your screen should then look like Figure 5.

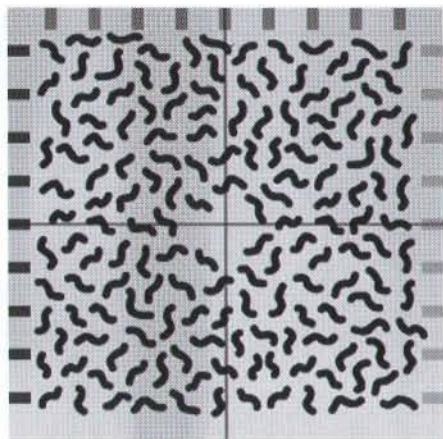


Figure 4. Lining up texture with edge markings and centre lines



Figure 5. The lining up texture activated.

CONCLUSION

In three articles we have covered quite a lot of ground. You should by now have a pretty good idea of how virtual reality systems based on multiple head-tracked displays such as CAVE and Cubby work. In the process you have learned quite a few nifty techniques. You have learned how to create and read an InputSprocket driver for an input device with three degrees of freedom. You now know how to support multiple views and how to mirror images. And if your matrix skills were a bit rusty, they should all be nice and shiny by now. Once you have a true 3D display such as Cubby the possibilities are virtually endless. We hope to be back with more information. In the meantime, please check out the Cubby web pages starting at <http://www.io.tudelft.nl/id-studiolab/cubby/index.html>.

REFERENCES

- Djajadiningrat, J.P. & Gribnau, M.W. (2000). *Cubby: Multiscreen Desktop VR Part I. Multiple views and mirroring images in QuickDraw 3D*. MacTech.
- Gribnau, M.W. & Djajadiningrat, J.P. (2000). *Cubby: Multiscreen Desktop VR Part II. How to create an InputSprocket driver for a 3D input device*. MacTech. 

<http://www.scientific.com>

Professional Software Developers

Looking for career opportunities?

Check out our website!

Nationwide Service

Employment Assistance

Resume Help

Marketability Assessment

Never a fee

Scientific Placement, Inc.

800-231-5920 800-757-9003 (Fax)

das@scientific.com

by Bob Boonstra, Westford, MA

CRUTCHES

Crutches? What an odd topic for a Programmer's Challenge, you might think. Let me explain. This month's problem was actually suggested by my wife, whose connection with the Challenge has until now been limited to the patience required to put up with the amount of time I spend running the contest. She recently had the misfortune to break her foot, which has, you guessed it, put her on crutches for six or so weeks. Being on crutches gives one a new perspective on distance, particularly distance between points around the house. And while she tries to stay off the foot as much as possible, she still has to get from place to place, so the broken foot also motivates one to find ways to minimize distance. Which leads us to this month's Challenge, a practical extension of the well-known Traveling Salesperson problem.

The prototype for the code you should write is:

```
typedef long Node;
typedef long Weight;

typedef struct Connection {
    Node node1; /* a connection exists between node1 ... */
    Node node2; /* ...and node2 ... */
    long distance; /* ...separated by this distance */
} Connection;

typedef struct Task {
    Weight weight; /* you need to carry an object with this weight ... */
    Node fromNode; /* ...from fromNode ... */
    Node toNode; /* ...to toNode */
} Task;

typedef enum {kPickUpObject=1, kDropOffObject, kMoveTo}
ActionType;

typedef struct Action {
    ActionType action; /* actions comprising the solution */
    long object; /* kPickUpObject or kDropOffObject this object */
    Node node /* kMoveTo this node */
} Action;

long /* actions in solution */ Crutches (
    const Node nodes[], /* Nodes defining the problem */
    long numNodes,
    const Connection connections[], /* Connections between nodes */
    long numConnections,
    const Task objectsToMove[], /* objects to be moved */
    long numObjects,
    Node startingNode, /* start from this node */
    Weight maxWeightToCarry, /* maximum weight that you can carry */
    Action solutionPath[] /* return your solution here */
);
```

Your job is to write code that will perform a set of Tasks and minimize the distance traveled in doing so. Each Task consists of moving an object of a specified weight from one

place (Node) to another. You can travel from one Node to another only if a Connection exists between the Nodes, and moving between a pair of Nodes requires traveling the associated distance along that Connection.

At the start of the problem, you are located at the startingNode. You are given the numNodes Nodes describing the problem space and the numConnections Connections between them. You are also given numObjects objectsToMove, each of which needs to be transported from the fromNode to the toNode. You can carry more than one object along your journey, provided the sum of the weights of the objects being carried does not exceed the maxWeightToCarry. The solution is described as a sequence of Actions. An Action consists of picking up an object (kPickUpObject) from the current Node, dropping off an Object at the current Node (kDropOffObject), or moving to an adjacent Node (kMoveTo) and carrying all objects that have been picked up to that Node. You may not pick up an object if doing so would cause the maxWeightToCarry to be exceeded. The sequence of Actions that transports all of the objectsToMove to the appropriate Nodes should be returned as the solutionPath, and Crutches should return the number of Actions in your solution.

None of the Tasks will be impossible to perform. No object will have a weight greater than maxWeightToCarry, so it will be possible to carry each object. It will be possible to reach each fromNode and each toNode by traversing Connections from the startingNode. Connections may not satisfy the triangle inequality, that is, it may be the case that a direct Connection between two Nodes is not the shortest path between them. No other a priori information about the Connections, Nodes, or Tasks is available.

Your solution will be evaluated first on correctness (as always), and then on score. Your score for this Challenge will be the total distance traveled to perform the required Tasks, plus a 10% penalty for each second of execution time expended. Lower scores are, of course, better.

The Challenge prize will be divided between the overall winner and the best scoring entry from a contestant that has not won the Challenge recently. If you have wanted to compete in the Challenge, but have been discouraged from doing so by the quality of the entries from our veteran contestants, perhaps this is your chance at some recognition and a share of the Challenge prize.

This will be a native PowerPC Challenge, using the CodeWarrior Pro 5 environment. Solutions may be coded in C, C++, or Pascal.

Next month, perhaps we'll solve the problem of how to motivate a couple of teenage children to perform these Tasks, allowing the broken foot more time to rest and heal. But perhaps that would be too difficult, even for Challenge readers. (Actually, the kids are being very helpful with the household chores.)

THREE MONTHS AGO WINNER

Congratulations to **Claes Wihlborg (Sweden)** for submitting the winning entry to the September Busy Beaver Challenge. This Challenge required contestants to do two things. First, contestants were to produce a 5-state "Busy Beaver" Turing Machine that writes as large a number of 1s as possible when given a blank input tape. Second, they had to write a general Turing Machine simulator that executes this Busy Beaver as quickly as possible. The problem statement provided a reference to a Turing Machine demonstrating that BB(5), the maximum number of 1s produced by any 5-state Busy Beaver, is at least 4098. Alas, none of the nine entries in this Challenge broke new ground in Busy Beaver research by providing a Busy Beaver that produced more than 4098 1s. So this competition was based on how quickly the Busy Beaver could be executed.

Claes' solution is extraordinarily fast, five times faster than the second place solution, and more than sixty times faster than the third-place solution. Upon investigation, while somewhat difficult to understand because of sparse commentary, Claes' entry is fascinating. To fully understand it, I inserted some debugging code and watched it in operation. I'll try to compensate for the terseness of the code by providing some additional explanation here.

The first thing Claes does is to call the `CreateOptimizedTMRules` routine to compile the Turing Machine rules into `OptimizedTMRules`. Two `OptimizedTMRules` are created for each Turing Machine rule, encoding both the rule and the associated binary input symbol. Each `OptimizedTMRule` contains a `OneBitActionRoutines` action field that combines two elements of the original Turing Machine rule: the move direction, and whether the output symbol is unchanged, 0, or 1. These actions are encoded as follows:

`abaLeft`, `abaRight`, `abaHalt` – leave the input unchanged and move left, right, or halt

`abaLeftSet`, `abaRightSet`, `abaHaltSet` – write a 1 and move left, right, or halt

`abaLeftClear`, `abaRightClear`, `abaHaltClear` – write a 0 and move left, right, or halt

This optimization allows Claes to factor out some logic tests during the Turing Machine simulation, and to avoid modifying the output tape when it doesn't change.

The `runNBitTuringMachine` performs the Turing Machine simulation. This routine processes the input tape in chunks of either 6 bits or 8 bits in size, trying the former first, and the latter if the former fails. It creates and uses two additional data structures, the `TapeSegment` structure that encodes a segment of the Turing Machine tape, and the `MacroNTMRule`

data structure that further encodes the `OptimizedTMRules`. The `TapeSegment` data structure takes advantage of the fact that repeating sequences can occur on a Turing Machine tape, and do occur on Busy Beaver Turing Machines. It contains a symbol field, which is a 6- or 8-bit section of the `tmTape`, an exponent that contains a repetition count for that section, and left and right pointers to adjacent tape segments.

The `MacroNTMRule` is a little more difficult to explain. The 256 Turing Machine states that the problem statement requires entries to support are expanded into 512 `OptimizedTMRules`, and further expand into 512×256 `MacroNTMRules`. A `MacroNTMRule` is indexed by 8 bits that identify the `OptimizedTMRule`, plus of the `TapeSegment` symbol value (6 or 8 bits). In effect, the `MacroNTMRule` expands the symbol set from 1 bit to 6 or 8 bits, and expands the set of Turing Machine states accordingly.

When `runNBitTuringMachine` executes the Turing Machine, it first looks to see if the `MacroNTMRule` corresponding to the current state has been created. If not, it calls the `createNbitMacroRule` routine to create it. This routine simulates the effect of the `OptimizedTMRules` on the current input symbol, determines what `newSymbol` output is produced, counts the number of `OptimizedTMRules` executed for this one `MacroNTMRule`, characterizes the nature of the rule into an `NBitActionRoutines` action, and stores a pointer to the new `MacroNTMRule` state.

The `NBitActionRoutines` field characterizes the `MacroNTMRule` by encoding the move direction, whether the output is different from the input, and whether the state is changed after processing this chunk of input. The most interesting values for this `NBitActionRoutines` field are as follows:

- `tbaBounceLeft`, `tbaBounceRight` – leave the input unchanged and reverse direction left or right
- `tTbaBounceLeftChange`, `tbaBounceRightChange` – write changed output and reverse direction left or right
- `tTbaThruLeft`, `tbaThruRight` – leave the input unchanged, continue moving left or right, and modify the state
- `tTbaThruLeftChange`, `tbaThruRightChange` – write changed output, continue moving left or right, and modify the state
- `tTbaThruOptimized`, `tbaThruRightOptimized` – leave the input unchanged, continue moving left or right, and stay in the same state
- `tTbaThruChangeOptimized`, `tbaThruRightChangeOptimized` – write changed output, continue moving left or right, and stay in the same state

The optimizations in the runNBitTuringMachine code allow the final output of Claes' Busy Beaver to be represented in a few TapeSegments:

Segment	Symbol (Hex)	Exponent (Hex)
0	0	180
1	1	1
2	36	1023
3	37	1
4	0	395

When runNBitTuringMachine returns, it indicates whether the attempt to execute with 6-bit tape segments succeeded or failed. If it failed, it is run again using 8-bit segments. While the 6-bit optimization is clearly intended to speed up the 5-state Busy Beaver Turing machines, it also kicks in for other machines. More importantly, the program meets the requirement of correctly simulating any Turing Machine with up to 256 states, without hard-coding any particulars of the Busy Beaver problem.

Before RunTuringMachine returns, it copies the TapeSegments back to the Turing Machine tape. (Until this point, the output of the Turing Machine exists only in the TapeSegment database.) For the 6-bit tape segment case, Claes uses an unrolled loop. For the 8-bit case, he uses memset. In both cases, the exponent field in the TapeSegment controls the number of times a TapeSegment symbol is copied.

I'd encourage you to take a look at Claes solution. I found it to be very clever.

The table below lists, for each of the solutions submitted, the number of 1s generated by the entry's Busy Beaver Turing Machine, the number of rules executed by that machine, the execution time of the Busy Beaver in milliseconds, and cumulative test case execution time. It also provides the code size, data size, and programming language used for each entry. As usual, the number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges prior to this one.

Name	# of 1s	# of Rules	BB Time (msecs)	Time (msecs)	Code Size	Data Size	Lang
Claes Wihlborg (9)	4098	11798826	0.93	3.04	6680	1080033	C
Ernst Munter (651)	4098	11798826	5.10	22.82	4764	749	C++
Mike Miller	4098	11798826	61.49	306.93	2044	410	C
Rob Shearer (51)	4098	11798826	64.92	311.59	1432	716	C++
Randy Boring (133)	4098	11798826	213.09	1066.58	4456	94	C++
Willeke Rieken (112)	4098	11798826	489.74	2456.34	912	16	C
Tom Saxton (165)	4098	11798826	734.27	3678.83	708	180	C++
Yung-Lueng Lan	4098	11798826	743.58	3726.49	1372	28	C
Ladislav Hala (7)	4098	47176870	2954.95	5922.87	1520	750	C



**A New Breed of
Application
Service
Provider**

Get

Your App



on the

WWWweb



Fast



www.thinkDog.com

**The Premier WebObjects
ASP**

TOP CONTESTANTS ...

Listed here are the Top Contestants for the Programmer's Challenge, including everyone who has accumulated 20 or more points during the past two years. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank Name		Points	Rank	Name	Points
1.	Munter, Ernst	231	6.	Shearer, Rob	48
2.	Saxton, Tom	106	7.	Taylor, Jonathan	36
3.	Maurer, Sebastian	68	8.	Wihlborg, Claes	29
4.	Rieken, Willeke	65	9.	Brown, Pat	20
5.	Boring, Randy	52			

... AND THE TOP CONTESTANTS LOOKING FOR A RECENT WIN

In order to give some recognition to other participants in the Challenge, we also list the high scores for contestants who have accumulated points without taking first place in a Challenge during the past two years. Listed here are all of those contestants who have accumulated 6 or more points during the past two years.

Rank	Name	Points	Rank	Name	Points
10.	Downs, Andrew	12	17.	Hala, Ladislav	7
11.	Jones, Dennis	12	18.	Miller, Mike	7
12.	Day, Mark	10	19.	Nicolle, Ludovic	7
13.	Duga, Brady	10	20.	Schotsman, Jan	7
14.	Fazekas, Miklos	10	21.	Widyatama, Yudhi	7
15.	Selengut, Jared	10	22.	Heithcock, JG	6
16.	Strout, Joe	10			

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place	20 points
2nd place	10 points
3rd place	7 points
4th place	4 points
5th place	2 points
finding bug	2 points
suggesting Challenge	2 points

Here is Claes' winning Busy Beaver solution:

BusyBeaver.c
Copyright © 2000
Claes Wihlborg

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "BusyBeaver.h"
```

```

BusyBeaver5
ulong /* return number of rules */ BusyBeaver5(
    TMRule theTMRules[]
    /* preallocated storage, return the rules for your BB machine */
)
{
    TMRule tm01[] = {
        {0,0, 1,1,kMoveRight},
        {0,1, 0,1,kMoveRight},
        {1,0, 2,1,kMoveLeft},
        {1,1, 1,1,kMoveLeft},
        {2,0, 0,1,kMoveRight},
        {2,1, 3,1,kMoveLeft},
        {3,0, 0,1,kMoveRight},
        {3,1, 4,1,kMoveLeft},
        {4,0, 1,1,kHalt},
        {4,1, 2,0,kMoveLeft}
    };

    memcpy( theTMRules, tm01, 10*sizeof(TMRule) );
    return 5*2;
}

```

```

typedef enum {obaNotYetDefined,
              obaLeft, obaLeftSet, obaLeftClear,
              obaHalt, obaHaltSet, obaHaltClear,
              obaRight, obaRightSet, obaRightClear }
OneBitActionRoutines;

typedef struct OptimizedTMRule {
    OneBitActionRoutines action;
    struct OptimizedTMRule *newState;
    /* set current state to newState when this rule fires */
} OptimizedTMRule;

typedef enum {tbaNotYetDefined, tbaUndefined,
              tbaHaltFromLeft, tbaHaltFromRight,
              tbaBounceLeft, tbaBounceRight,
              tbaBounceLeftChange, tbaBounceRightChange,
              tbaThruLeft, tbaThruRight,
              tbaThruLeftChange, tbaThruRightChange,
              tbaThruLeftOptimized, tbaThruRightOptimized,
              tbaThruLeftChangeOptimized,
              tbaThruRightChangeOptimized } NBitActionRoutines;

typedef struct MacroNTMRule {
    NBitActionRoutines action;
    unsigned char newSymbol;
    unsigned short ruleCount;
    struct MacroNTMRule *newState;
    /* set current state to newState when this rule fires */
} MacroNTMRule;

typedef struct TapeSegment {
    unsigned int symbol;
    unsigned int exponent;
    struct TapeSegment *left,*right;
} TapeSegment;

```

```

static OptimizedTMRule optimizedTMRules[512];
static unsigned int highestState;

static MacroNTMRule rules[256*2*256];

static unsigned int nBit, maxBit;

#define tapeDim 1520
static int nxtTapeIx;
static TapeSegment myTape[tapeDim]. *freeSegment,
*rightmostSegment, *leftmostSegment;

static unsigned char *TheTapeCenter;
static ulong allocatedLeft, maxALeft, allocatedRight,
maxARight;
static ulong numberOfIsOnInputTape;

static int aLotOfZeroes[] = {
    0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
    0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
    0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0,
    0,0,0,0,0,

```

```

0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0,
0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0,
0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0,
0,0,0,0,0  );

```

FUNCTION PROTOTYPES

```

static Boolean CreateOptimizedTMRules(
    TMRule theTMRules[], /* contains the rules for your BB machine */
    ulong numberOfTMRules);

static void createNbitMacroRule( unsigned int inSymbol,
    MacroNTMRule *inState );

static Boolean MoreSegments( void );
static TapeSegment *GetFreeSegment( void );
static TapeSegment *expandLeft( void );
static TapeSegment *expandRight( void );

static Boolean runNbitTuringMachine(
    ulong *numberOfIsGenerated,
    ulong *numberOfRulesExecuted
);

```

CreateOptimizedTMRules

```

static Boolean CreateOptimizedTMRules(
    TMRule theTMRules[], /* contains the rules for your BB machine */
    ulong numberOfTMRules)
{
    unsigned int i,state,inputSymbol;
    OptimizedTMRule *destRule;

    highestState = 0;
    memset( optimizedTMRules, 0x00,
        512*sizeof(OptimizedTMRule) );

    for(i=0; i<numberOfTMRules; i++)
    {
        if ((state = theTMRules[i].oldState) > highestState)
            highestState = state;
        destRule = optimizedTMRules + ((theTMRules[i].oldState
        << 1) +
        (inputSymbol=theTMRules[i].inputSymbol));
        if (destRule->action != obaNotYetDefined)
        {
            printf("State: %d Input: %d multiply defined\n",
                theTMRules[i].oldState,inputSymbol);
            return false;
        }
        destRule->newState = optimizedTMRules +
            (theTMRules[i].newState
        << 1);
        destRule->action = obaHalt +
        3*theTMRules[i].moveDirection;
        if (inputSymbol != theTMRules[i].outputSymbol)
            if (theTMRules[i].inputSymbol) destRule-
        >action+=2;
        else destRule->action++;
    }
    return true;
}

```

createNbitMacroRule

```

static void createNbitMacroRule( unsigned int inSymbol,
    MacroNTMRule *inState )
{
    int mySymbol = inSymbol;
    int bit;
    OptimizedTMRule *state;
    int iState;
    OneBitActionRoutines action;
    int ruleCount = 0;
    MoveDir inDir,outDir;

    iState = (inState - rules) >> nBit;
    state = optimizedTMRules + (iState & 0xfffe);

    if (iState & 1)
    {
        bit = 1;

```

```

        inDir = kMoveLeft;
    }
    else
    {
        bit = maxBit;
        inDir = kMoveRight;
    }

```

```

loop:
    ruleCount++;
    if (mySymbol & bit)
    {
        action = state[1].action;
        state = state[1].newState;
    }
    else
    {
        action = state->action;
        state = state->newState;
    }
    switch (action)
    {
        case obaNotYetDefined:
            inState->ruleCount = ruleCount;
            inState->newSymbol = mySymbol;
            inState->action = tbaUndefined;
            return;
        case obaLeft:
            if ((bit <= 1) <= maxBit) goto loop;
            break;
        case obaLeftSet:
            mySymbol |= bit;
            if ((bit <= 1) <= maxBit) goto loop;
            break;
        case obaLeftClear:
            mySymbol &= ~1 - bit;
            if ((bit <= 1) <= maxBit) goto loop;
            break;
        case obaHalt:
            inState->ruleCount = ruleCount;
            inState->newSymbol = mySymbol;
            inState->action = (inDir == kMoveRight)?
                tbaHaltFromLeft : tbaHaltFromRight;
            return;
        case obaHaltSet:
            mySymbol |= bit;
            inState->ruleCount = ruleCount;
            inState->newSymbol = mySymbol;
            inState->action = (inDir == kMoveRight)?
                tbaHaltFromLeft : tbaHaltFromRight;
            return;
        case obaHaltClear:
            mySymbol &= ~1 - bit;
            inState->ruleCount = ruleCount;
            inState->newSymbol = mySymbol;
            inState->action = (inDir == kMoveRight)?
                tbaHaltFromLeft : tbaHaltFromRight;
            return;
        case obaRight:
            if (bit >= 1) goto loop;
            break;
        case obaRightSet:
            mySymbol |= bit;
            if (bit >= 1) goto loop;
            break;
        case obaRightClear:
            mySymbol &= ~1 - bit;
            if (bit >= 1) goto loop;
            break;
    }

    inState->ruleCount = ruleCount;
    inState->newSymbol = mySymbol;
    inState->action = tbaBounceLeft;
    if (!bit)
    {
        outDir = kMoveRight;
        inState->action++;
    }
    else
    {
        outDir = kMoveLeft;
        state++;
    }
}

```

```

if (mySymbol != inSymbol)
    inState->action += 2;

inState->newState = ((state - optimizedTMRules) << nBit) +
                    rules;
if (outDir == inDir)
{
    inState->action += 4;
    if ((state - optimizedTMRules) == iState)
        inState->action += 4;
}

return;
}

```

MoreSegments

```

static Boolean MoreSegments( void )
{
    int end;
    if (nBit == 6) return false;
    if (nxtTapeIx < tapeDim)
    {
        end = nxtTapeIx + 149;
        freeSegment = myTape + nxtTapeIx;
        do
        {
            myTape[nxtTapeIx].right = myTape + nxtTapeIx + 1;
        }
        while (++nxtTapeIx < end);
        myTape[nxtTapeIx].right = 0;
        return true;
    }
    // printf("Segments finito!!!\n");
    return false;
}

```

GetFreeSegment

```

static TapeSegment *GetFreeSegment( void )
{
    TapeSegment *tmp;

    if (freeSegment || MoreSegments())
    {
        tmp = freeSegment;
        freeSegment = tmp->right;
        return tmp;
    }
    return 0;
}

```

expandLeft

```

static TapeSegment *expandLeft( void )
{
    ulong delta;
    TapeSegment *tmp, *newCurrent, *oldLeftmost;
    unsigned char *p, *pOld;
    ulong oneCount, oldValue;

    if (!(delta = (((maxALeft - allocatedLeft) > 300) ?
                    300 : (maxALeft - allocatedLeft))))
        return 0;

    allocatedLeft += delta;
    p = TheTapeCenter - allocatedLeft;

    if (!(newCurrent = GetFreeSegment()))
        return 0;
    newCurrent->left = 0;
    oldLeftmost = leftmostSegment;
    leftmostSegment = newCurrent;

    if (!memcmp( p, aLotOfZeroes, delta))
    {
        newCurrent->symbol = 0;
        newCurrent->exponent = 8*delta / nBit;
    }
    else
    if (nBit == 8)
    {
        newCurrent->symbol = *p;
        newCurrent->exponent = 1;
    }
}

```

```

pOld = p + delta;
while (++p < pOld)
{
    if (*p == newCurrent->symbol)
    {
        newCurrent->exponent++;
    }
    else
    {
        tmp = newCurrent;
        if (!(newCurrent = GetFreeSegment()))
            return 0;
        newCurrent->left = tmp;
        tmp->right = newCurrent;
        newCurrent->symbol = *p;
        newCurrent->exponent = 1;
    }
}
tmp = newCurrent;
do
{
    if ((oldValue = tmp->symbol) != 0)
    {
        oneCount = 0;
        do {
            ++oneCount;
        } while (oldValue = oldValue & (oldValue-1));
        numberOfIsOnInputTape += oneCount*tmp->exponent;
    }
    while (tmp = tmp->left);
}
else /* nBit == 6 */
{
    return 0;
}

newCurrent->right = oldLeftmost;
if (oldLeftmost)
{
    oldLeftmost->left = newCurrent;
}
else
{
    rightmostSegment = newCurrent;
}
return newCurrent;
}

```

expandRight

```

static TapeSegment *expandRight( void )
{
    ulong delta;
    TapeSegment *tmp, *newCurrent, *oldRightmost;
    unsigned char *p, *pOld;
    ulong oneCount, oldValue;

    if (!(delta = (((maxARight - allocatedRight) > 300) ?
                    300 : (maxARight - allocatedRight))))
        return 0;

    pOld = TheTapeCenter + allocatedRight;
    allocatedRight += delta;

    if (!(newCurrent = GetFreeSegment()))
        return 0;
    newCurrent->right = 0;
    oldRightmost = rightmostSegment;
    rightmostSegment = newCurrent;

    if (!memcmp( pOld, aLotOfZeroes, delta))
    {
        newCurrent->symbol = 0;
        newCurrent->exponent = 8*delta / nBit;
    }
    else
    if (nBit == 8)
    {
        p = pOld + delta - 1;
        newCurrent->symbol = *p;
        newCurrent->exponent = 1;
        while (-p >= pOld)
        {

```



OPENBASE[®] SQL

6.5

DATABASE ENGINE

OpenBase SQL 6.5

The high-performance
Database for MacOS X

OPENBASE FEATURES

- JDBC Driver
- WebObjects Adaptors
- REALbasic Plugin
- Multi-Threaded Server
- Row Level Locking
- 100 MB Searchable Blobs
- Database Replication
- Database Management Tools
- Graphical Schema Design Tools
- Remote Administration

We process over 2 million
transactions each day.

OpenBase performance has
been outstanding."

FlightArrivals.com

AMAZING SPEED

AMAZING POWER

[AMAZINGLY AFFORDABLE]

With over 2 Million Transactions per day, **FlightArrivals.com** flies with **OpenBase SQL**

Take OpenBase SQL for a Test Flight Today!

Get Your FREE Developer's License at www.openbase.com



OPENBASE[®]
INTERNATIONAL

```

if (*p == newCurrent->symbol)
{
    newCurrent->exponent++;
}
else
{
    tmp = newCurrent;
    if (!(newCurrent = GetFreeSegment()))
        return 0;
    newCurrent->right = tmp;
    tmp->left = newCurrent;
    newCurrent->symbol = *p;
    newCurrent->exponent = 1;
}
}
tmp = newCurrent;
do
{
    if ((oldValue = tmp->symbol) != 0)
    {
        oneCount = 0;
        do {
            ++oneCount;
        } while (oldValue = oldValue & (oldValue-1));
        numberOfIsOnInputTape += oneCount*tmp->exponent;
    }
    while (tmp = tmp->right);
} else /*nBit==6*/
{
    return 0;
}

newCurrent->left = oldRightmost;
oldRightmost->right = newCurrent;
return newCurrent;
}

```

```

static Boolean runNBitTuringMachine(
    ulong *numberOfIsGenerated,
    ulong *numberOfRulesExecuted
)
{
    // Local data areas
    ulong oneCount;
    ulong ruleCount = 0;
    int oldValue,i;
    MacroNTMRule *state;
    TapeSegment *currentSegment, *tmp;
    Boolean resultat = false;

    // Init

    *numberOfIsGenerated = 0;
    numberOfIsOnInputTape = 0;

    memset( rules, 0x00,
(highestState+1)*2*(1<<nBit)*sizeof(MacroNTMRule) );
    state = rules;

    for (i=0;i<19;i++)
    {
        myTape[i].right = myTape + i + 1;
    }
    myTape[19].right = 0;
    freeSegment = myTape;

    rightmostSegment = 0;
    leftmostSegment = 0;
    allocatedLeft = 0;
    allocatedRight = 0;

    expandLeft();
    currentSegment = expandRight();
}
runNBitTuringMachine

```

```

mainLoop:
    state += currentSegment->symbol;
loop:
    ruleCount += state->ruleCount;
    switch (state->action)
    {
    case tbaNotYetDefined:
        createNbitMacroRule( currentSegment->symbol, state );
        goto loop;

    case tbaUndefined:
        goto avsluta;

    case tbaHaltFromLeft:
        if (currentSegment->exponent > 1)
        {
            tmp = currentSegment;
            if (!(currentSegment = freeSegment))
            {
                if (!MoreSegments()) goto avsluta;
                currentSegment = freeSegment;
            }
            freeSegment = currentSegment->right;
            currentSegment->left = tmp->left;
            currentSegment->left->right = currentSegment;
            currentSegment->right = tmp;
            tmp->left = currentSegment;
            currentSegment->exponent = 1;
            tmp->exponent--;
        }
        currentSegment->symbol = state->newSymbol;
        break;

    case tbaHaltFromRight:
        if (currentSegment->exponent > 1)
        {
            tmp = currentSegment;
            if (!(currentSegment = freeSegment))
            {
                if (!MoreSegments()) goto avsluta;
                currentSegment = freeSegment;
            }
            freeSegment = currentSegment->right;
            currentSegment->left = tmp;
            currentSegment->right = tmp->right;
            tmp->right->left = currentSegment;
            tmp->right = currentSegment;
            currentSegment->exponent = 1;
            tmp->exponent--;
        }
        currentSegment->symbol = state->newSymbol;
        break;

    case tbaBounceLeft:
        state = state->newState;
        currentSegment = currentSegment->left;
        goto mainLoop;

    case tbaBounceRight:
        currentSegment = currentSegment->right;
        state = state->newState;
        goto mainLoop;

    case tbaBounceLeftChange:
        if (currentSegment->exponent > 1)
        {
            tmp = currentSegment;
            if (!(currentSegment = freeSegment))
            {
                if (!MoreSegments()) goto avsluta;
                currentSegment = freeSegment;
            }
            freeSegment = currentSegment->right;
            currentSegment->left = tmp->left;
            currentSegment->right = tmp;
            tmp->left->right = currentSegment;
            tmp->left = currentSegment;
            currentSegment->exponent = 1;
            tmp->exponent--;
        }
        currentSegment->symbol = state->newSymbol;
        state = state->newState;
        currentSegment = currentSegment->left;
        goto mainLoop;

    case tbaBounceRightChange:
        if (currentSegment->exponent > 1)
        {
            tmp = currentSegment;
            if (!(currentSegment = freeSegment))
            {
                if (!MoreSegments()) goto avsluta;
                currentSegment = freeSegment;
            }
            freeSegment = currentSegment->right;
            currentSegment->left = tmp;
            currentSegment->right = tmp->right;
            tmp->right->left = currentSegment;
            tmp->right = currentSegment;
            currentSegment->exponent = 1;
            tmp->exponent--;
        }
        currentSegment->symbol = state->newSymbol;
        currentSegment = currentSegment->right;
        state = state->newState;
        goto mainLoop;

    case tbaThruLeft:
        if (currentSegment->exponent > 1)
        {
            tmp = currentSegment;
            if (!(currentSegment = freeSegment))
            {
                if (!MoreSegments()) goto avsluta;
                currentSegment = freeSegment;
            }
            freeSegment = currentSegment->right;
            currentSegment->left = tmp;
            currentSegment->right = tmp->right;
            tmp->right->left = currentSegment;
            tmp->right = currentSegment;
            currentSegment->exponent = 1;
            tmp->exponent--;
        }
        state = state->newState;
        if (currentSegment = currentSegment->left)
            goto mainLoop;
        if (currentSegment = expandLeft()) goto mainLoop;
        break;

    case tbaThruRight:
        if (currentSegment->exponent > 1)
        {
            tmp = currentSegment;
            if (!(currentSegment = freeSegment))
            {
                if (!MoreSegments()) goto avsluta;
                currentSegment = freeSegment;
            }
            freeSegment = currentSegment->right;
            currentSegment->left = tmp->left;
            currentSegment->right = tmp;
            tmp->left->right = currentSegment;
            tmp->left = currentSegment;
            currentSegment->exponent = 1;
            tmp->exponent--;
        }
        state = state->newState;
        if (currentSegment = currentSegment->right)
            goto mainLoop;
        if (currentSegment = expandRight()) goto mainLoop;
        break;

    case tbaThruLeftChange:
        if (currentSegment->exponent > 1)
        {
            tmp = currentSegment;
            if (!(currentSegment = freeSegment))
            {
                if (!MoreSegments()) goto avsluta;
                currentSegment = freeSegment;
            }
            freeSegment = currentSegment->right;
            currentSegment->left = tmp;
            currentSegment->right = tmp->right;
            tmp->right->left = currentSegment;
            tmp->right = currentSegment;
            currentSegment->exponent = 1;
            tmp->exponent--;
        }

```

```

    }
    currentSegment->symbol = state->newSymbol;
    state = state->newState;
    if (currentSegment = currentSegment->left)
        goto mainLoop;
    if (currentSegment = expandLeft()) goto mainLoop;
    break;

case tbaThruRightChange:
    if (currentSegment->exponent > 1)
    {
        tmp = currentSegment;
        if (!(currentSegment = freeSegment))
        {
            if (!MoreSegments()) goto avsluta;
            currentSegment = freeSegment;
        }
        freeSegment = currentSegment->right;
        currentSegment->left = tmp->left;
        currentSegment->right = tmp;
        tmp->left->right = currentSegment;
        tmp->left = currentSegment;
        currentSegment->exponent = 1;
        tmp->exponent--;
    }
    currentSegment->symbol = state->newSymbol;
    state = state->newState;
    if (currentSegment = currentSegment->right)
        goto mainLoop;
    if (currentSegment = expandRight()) goto mainLoop;
    break;

case tbaThruLeftOptimized:
    while ((tmp = currentSegment->left) &&
        (currentSegment->symbol == tmp->symbol))
    {
        currentSegment->exponent += tmp->exponent;
        tmp->right = freeSegment;
        freeSegment = tmp;
        currentSegment->left = tmp->left;
        if (currentSegment->left)
            currentSegment->left->right = currentSegment;
        else
            leftmostSegment = currentSegment;
    }

```

```

        ruleCount += state->ruleCount*
            (currentSegment->exponent-
1);
        state = state->newState;
        if (currentSegment = currentSegment->left)
            goto mainLoop;
        if (currentSegment = expandLeft()) goto
mainLoop;
        break;

case tbaThruRightOptimized:
    while ((tmp = currentSegment->right) &&
        (currentSegment->symbol == tmp-
>symbol))
    {
        currentSegment->exponent += tmp->exponent;
        currentSegment->right = tmp->right;
        tmp->right = freeSegment;
        freeSegment = tmp;
        if (currentSegment->right)
            currentSegment->right->left =
currentSegment;
        else
            rightmostSegment = currentSegment;
    }
    ruleCount += state->ruleCount*
        (currentSegment->exponent-
1);
    state = state->newState;
    if (currentSegment = currentSegment->right)
        goto mainLoop;
    if (currentSegment = expandRight()) goto
mainLoop;
    break;

case tbaThruLeftChangeOptimized:
    while ((tmp = currentSegment->left) &&
        (currentSegment->symbol == tmp-
>symbol))
    {
        currentSegment->exponent += tmp->exponent;
        tmp->right = freeSegment;
        freeSegment = tmp;
        currentSegment->left = tmp->left;
        if (currentSegment->left)

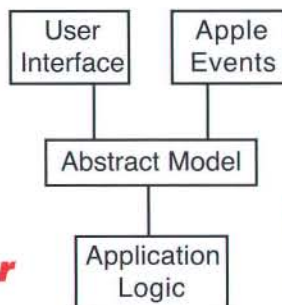
```

Fight Boredom

Let's face it: Much of programming is boring and repetitive. Well, that's where the right tool can save days, weeks, or even months of your valuable time.

AppMaker Your Assistant Programmer

AppMaker makes it faster and easier to make an application. It's like having your own assistant programmer. You point and click to tell AppMaker the results you want, then it generates "human, professional quality code" to implement your design.



Model-View-Controller

AppMaker's generated code uses the MVC paradigm. It separates the user interface from application logic, making code easier to write. You deal only with abstract data; AppMaker takes care of the user interface.

Scriptable Applications

AppMaker generates the 'aete' resource and generates code to access your data (Properties and Elements in the Apple Event Object Model) and to handle Events.

Just \$199 from www.devdepot.com

B•O•W•E•R•S
Development

P.O. Box 929, Grantham, NH 03753 • (603) 863-0945 • FAX 863-3857
bowersdev@aol.com • <http://members.aol.com/bowersdev>

```

        currentSegment->left->right = currentSegment;
        else
            leftmostSegment = currentSegment;
    }
    currentSegment->symbol = state->newSymbol;
    ruleCount += state->ruleCount *
        (currentSegment->exponent-1);
    state = state->newState;
    if (currentSegment = currentSegment->left)
        goto mainLoop;
    goto mainLoop;
if (currentSegment = expandLeft()) goto mainLoop;
break;

case tbaThruRightChangeOptimized:
    while ((tmp = currentSegment->right) &&
        (currentSegment->symbol == tmp->symbol))
    {
        currentSegment->exponent += tmp->exponent;
        currentSegment->right = tmp->right;
        tmp->right = freeSegment;
        freeSegment = tmp;
        if (currentSegment->right)
            currentSegment->right->left = currentSegment;
        else
            rightmostSegment = currentSegment;
    }
    currentSegment->symbol = state->newSymbol;
    ruleCount += state->ruleCount *
        (currentSegment->exponent-1);
    state = state->newState;
    if (currentSegment = currentSegment->right) goto mainLoop;
    if (currentSegment = expandRight()) goto mainLoop;
}

if (currentSegment)
{
    resultat = true;
}

avsluta:

*numberOfRulesExecuted = ruleCount;

for (currentSegment = leftmostSegment; currentSegment;
    currentSegment=currentSegment->right)
{
    if ((oldValue = currentSegment->symbol)!=0)
    {
        oneCount = 0;
        do {
            ++oneCount;
        } while (oldValue = oldValue & (oldValue-1));
        *numberOfIsGenerated += oneCount*currentSegment->exponent;
    }
}

// bug in following line corrected by JRB, doesn't affect results
// numberOfIsGenerated -= numberOfIsOnInputTape;
*numberOfIsGenerated -= numberOfIsOnInputTape;

return resultat;
}

RunTuringMachine
Boolean /* return true for success */ RunTuringMachine(
    TMRule theTMRules[],
    /* contains the rules for your BB machine */
    ulong numberOfTMRules,
    /* the number of rules in theTMRules */
    ulong numBytesInHalfTape,
    /* half-size of the "infinite" Turing Machine tape */
    unsigned char *tmTape,
    /* pointer to preallocated Turing Machine tape storage */
    /* Each byte contains 8 tape symbols, each symbol is 0 or 1. */
    /* The tape extends from tmTape[numBytesInHalfTape] to
        tmTape[numBytesInHalfTape-1] */
    /* Tape position 0 is (tmTape[0] & 0x80),
        tape position 1 is (tmTape[0] & 0x40)
        tape position -1 is (tmTape[-1] & 0x01), etc. */
    ulong *numberOfIsGenerated,
    /* return the number of 1s placed on the tape */
    ulong *numberOfRulesExecuted
    /* return the number of rules executed when running BB, including the halt rule */
)

```

```

{
// Local data areas
    unsigned char *p;
    ulong myDoubleSymbol;
    Boolean resultat;
    TapeSegment *currentSegment;

// Init data areas
    if (!CreateOptimizedTMRules( theTMRules, numberOfTMRules))
    {
        return false;
    }

    TheTapeCenter = tmTape;

    nBit = 6;
    maxBit = 0x20;

    maxALeft = numBytesInHalfTape - numBytesInHalfTape%3;
    maxARight = maxALeft;

    resultat = runNBitTuringMachine( numberOfIsGenerated,
        numberOfRulesExecuted );

    if (resultat)
    {
        p = tmTape - allocatedLeft;
        currentSegment = leftmostSegment;

#define CHECK_SEGMENT \
        if (-(currentSegment->exponent) == 0) \
            currentSegment=currentSegment->right

        while (currentSegment)
        {
            myDoubleSymbol = currentSegment->symbol << 18;
            CHECK_SEGMENT;
            myDoubleSymbol |= currentSegment->symbol << 12;
            CHECK_SEGMENT;
            myDoubleSymbol |= currentSegment->symbol << 6;
            CHECK_SEGMENT;
            myDoubleSymbol |= currentSegment->symbol;
            CHECK_SEGMENT;
            *p++ = myDoubleSymbol>>16;
            *p++ = myDoubleSymbol>>8;
            *p++ = myDoubleSymbol;
        }

        return true;
    }

    nBit = 8;
    maxBit = 0x80;

    maxALeft = numBytesInHalfTape;
    maxARight = maxALeft;
    nxtTapeIx = 20;

    resultat = runNBitTuringMachine( numberOfIsGenerated,
        numberOfRulesExecuted );

    p = tmTape - allocatedLeft;
    for (currentSegment = leftmostSegment; currentSegment;
        currentSegment=currentSegment->right)
    {
        memset( p, currentSegment->symbol, currentSegment-
            >exponent );
        p += currentSegment->exponent;
    }

    return resultat;
}

```

by Tim Monroe

Timecode

Using QuickTime's Timecode Media Handler

INTRODUCTION

It's often useful — especially when editing some video or audio media — to be able to assign a precise time stamp to a place in a stream of media data. In 1967, the Society of Motion Picture and Television Engineers (SMPTE) introduced a method for doing this, called timecoding. *Timecoding* is a way of putting time stamps (called *timecodes*) on recorded media such as film, video, sound, and the like.

QuickTime, of course, has its own methods of specifying times within a movie, using time bases, time values, and time scales. (See "Movie Controller Potpourri" in *MacTech*, February, 1999 for a preliminary discussion of these concepts.) But QuickTime also supports storing and displaying SMPTE timecodes. **Figure 1** shows a frame of a QuickTime movie with a timecode value displayed below the video track.

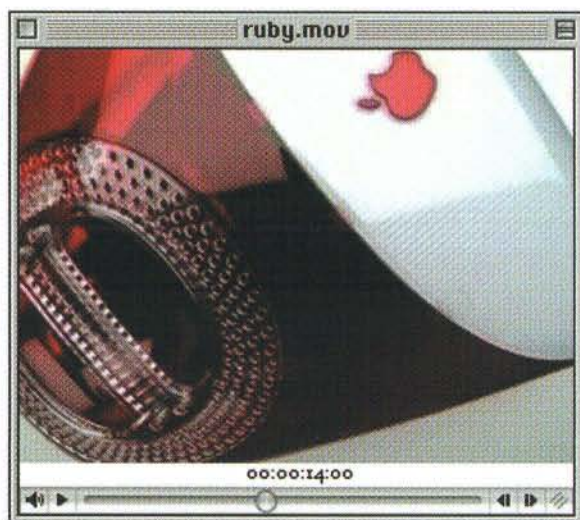


Figure 1. A movie frame with timecode displayed.

Timecode is displayed using an eight-digit, 24-hour clock that looks like this: *hh:mm:ss:ff*. The digits represent (from left to right) hours, minutes, seconds, and frames. As you can see, the frame displayed in **Figure 1** is exactly 14 seconds from the beginning of the movie (assuming that the movie began at timecode 00:00:00:00). The actual timecode values of a specific QuickTime movie can be synthesized by the computer or captured along with the source material.

In this article, we'll take a look at using timecodes in QuickTime movies. Timecodes are stored in timecode tracks, which are created and managed using the timecode media handler. The timecode media handler is extremely easy to work with; it provides a grand total of 11 functions (of which we'll use only 7 here) and requires us to work with only a handful of data structures. In fact, there is only one mildly difficult concept we'll need to understand: the notion of drop-frame timecode. We'll begin by investigating the standard timecode formats, and then we'll move on to seeing how to create and manage timecode tracks in QuickTime movies.

Our sample application this month is called *QTTimeCode*; its Test menu contains the timecode-specific commands. **Figure 2** shows the Test menu for movies that do not contain a timecode track.

Test	
Add Timecode Track...	⌘1
Remove Timecode Tracks	⌘2
Show Current Timecode	⌘3
Show Timecode Source	⌘4
Toggle Timecode Display	⌘5

Figure 2. The Test menu of *QTTimeCode*.

Tim Monroe is happy to announce the birth of a baby lizard. *Libra*, the son or daughter of Sean and Avril, was born in early October. You can send reptile layette items to him at monroe@apple.com.

Figure 3 shows the Test menu for movies that do contain a timecode track.

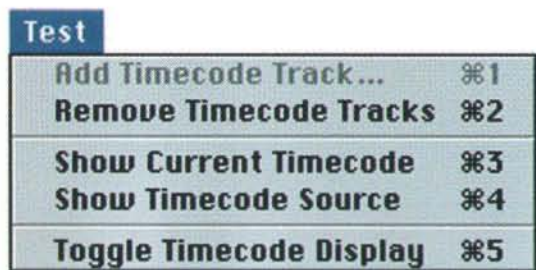


Figure 3. The Test menu of QTTimeCode.

TIMECODE STANDARDS

As we've seen, timecode is expressed using an eight-digit clock, where the rightmost two digits represent a frame number. There are four standard frame rates used commercially around the world, so there are four standard timecode formats:

- Most film produced in the United States uses a standard of 24 frames per second (fps).
- Most film produced in Europe and Australia uses a standard of 25 fps, established by the European Broadcasting Union (EBU); 25 fps is also used for PAL or SECAM video and television.
- The SMPTE standard for American black-and-white television uses 30 fps. This frame rate is also commonly used for audio and CD mastering.
- American color television and systems based on it (such as NTSC video) use a standard of 29.97 fps. This rate is the result of certain technical decisions made when converting from black-and-white to color television.

The oddball here is the SMPTE standard for American color television and NTSC video, which uses a frame rate of 29.97. This is close enough to 30 that people typically consider American color TV to have a frame rate of 30 fps. But 29.97 isn't *exactly* 30, and the difference can add up. For instance, over a period of one hour, a 30 fps movie will display 108,000 frames, whereas a 29.97 fps movie will display 107,892 frames. This means that, if we use a 30 fps timecode to display the time of a 29.97 fps movie, the movie would be 108 frames (or about 3.6 seconds) longer than a movie that has exactly 30 fps. That is to say, by the time a 29.97 fps movie gets to 108,000 frames, 1 hour and 3.6 seconds has elapsed.

For most purposes, an extra 3.6 seconds per hour is pretty insignificant. But on commercial TV, time is money. For example, ABC Broadcasting is expected to charge

\$2.2 million for a 30-second advertisement during the 2001 Super Bowl. In that case, 3.6 seconds is worth about \$264,000. So it's sometimes important to get timecode values exactly right.

To ensure that 30 fps timecodes displayed by a 29.97 fps movie exactly match the time on a wall clock, SMPTE defined *drop-frame timecoding*, according to which two frame numbers are dropped at the start of every minute, except for every minute that is evenly divisible by ten. (For example, the timecode value 11:31:59:29 is immediately followed by 11:32:00:02.) So, over the course of one hour, 108 frame numbers will be dropped, which is exactly what is needed to bring the 30 fps timecode display into synchronization with a wall clock. In spite of the name ("drop-frame timecode"), no video frames are dropped. Rather, frame *numbers* are dropped from the timecode display so that the timecode stays in synch with a wall clock.

When drop-frame timecode is being used, it's conventional to use a comma (',') or a semicolon (';') to separate the seconds digits from the frames digits. **Figure 4** shows a QuickTime movie that uses drop-frame timecode.



Figure 4. A movie frame with drop-frame timecode.

Keep in mind that in a timecode, the rightmost two digits represent a frame number, not (say) hundredths of a second. This is worth remarking, if only because there are occasions where QuickTime does use the rightmost digits to represent fractions of a second. For instance, in the previous *QuickTime Toolkit* article ("Word Is Out" in *MacTech*, November 2000), we saw that the text movie importer recognizes a number of text descriptors, one of which is a timestamp indicating the time at which the imported text sample is to begin. In this case, the rightmost digits represent a number of time scale units.



Programming Doesn't Have To Be This Difficult. It's REALbasic!

NEW
Version!
REALbasic 2.1*
FREE
Update!

REALbasic is the award-winning, visual, object-oriented BASIC development environment for the Macintosh.

Use REALbasic's visual interface builder and platform-independent language to build native, compiled—not interpreted—professional quality applications in a fraction of the time it would take in C/C++. Because our language is platform-independent you only need to write a single set of code to create applications for both Macintosh and Windows. Leverage your C/C++ experience to extend REALbasic's capabilities using shared libraries, Mac OS Toolbox and Win32 API calls or by writing cross-platform REALbasic plug-ins. Create prototypes, Internet applications, database front-ends, even games with REALbasic! Its OOP language employs everything

you'd expect from a modern development environment—methods, properties, classes, subclassing, inheritance, constructors and destructors, virtual methods, and more. The IDE supports multi-threading, extensibility, TCP/IP controls, plus multimedia and QuickTime tools, and support for standards such as SQL, ODBC, AppleScript, and XCMDs. You can even import Visual Basic forms and modules.

Go to www.realbasic.com NOW to download a FREE trial version or call 512.263.1233.





2000 Runner-Up - Best Macintosh User Experience

*Free update for all owners of REALbasic 2.0 and above. REALbasic and the REALbasic logo are trademarks of REAL Software, Inc. Apple and the Apple logo are trademarks of Apple Computer, Inc., registered in the U.S., used with permission. All other trademarks are the property of their respective owners.

TIMECODE IN QUICKTIME

QuickTime version 2.0 introduced the *timecode media handler*, which we can use to create timecode tracks in movies, hide and show those tracks, get and set information about timecode tracks, and so forth. The timecode media handler supports both drop-frame and non-drop-frame timecodes, as well as a simple counter (shown in **Figure 5**).

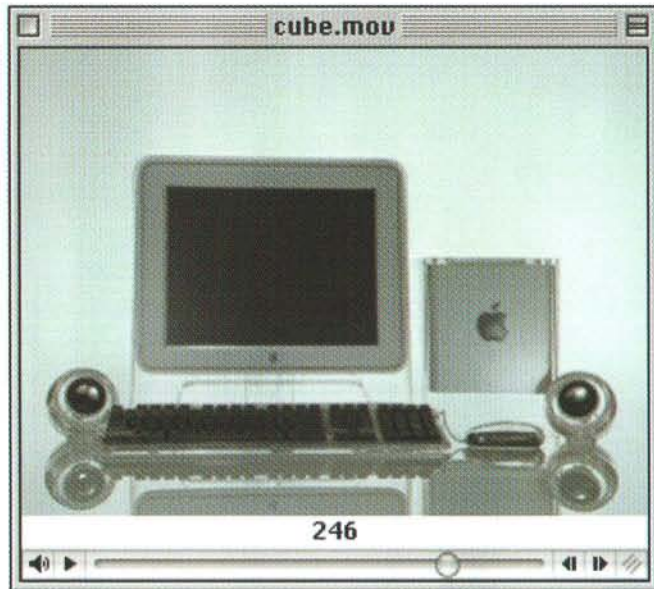


Figure 5. A movie frame with a counter.

A timecode track specifies timing information for one or more other tracks, which are typically video or sound tracks. To associate the timecode track with a target track, the target track contains a track reference to the timecode track. (See the “Word Is Out” article once again for a discussion of track references.)

All of the QuickTime functions for working with timecodes take an instance of the timecode media handler as a parameter. Our sample application QTTimeCode creates a single timecode track in a movie, so we can call `GetMovieIndTrackType` and `GetMediaHandler` to find a movie’s timecode media handler, as shown in Listing 1.

Listing 1: Finding a movie’s timecode media handler instance

```

QTTC_GetTimeCodeMediaHandler
MediaHandler QTTC_GetTimeCodeMediaHandler (Movie
theMovie)
{
    Track        myTrack = NULL;
    Media        myMedia = NULL;
    MediaHandler myHandler = NULL;

    // get the (first) timecode track in the specified movie
    myTrack = GetMovieIndTrackType(theMovie, 1,
        TimeCodeMediaType, movieTrackMediaType);
    if (myTrack != NULL) {

```

```

        // get the timecode track's media and media handler
        myMedia = GetTrackMedia(myTrack);
        if (myMedia != NULL)
            myHandler = GetMediaHandler(myMedia);
    }
    return(myHandler);
}

```

When the user selects the “Add Timecode Track...” command in the Test menu of our QTTimeCode application, QTTimeCode displays the dialog box shown in **Figure 6** to elicit some settings from the user.

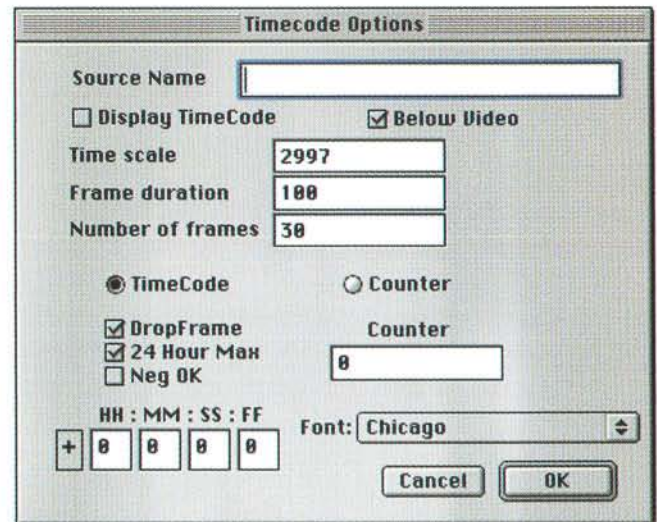


Figure 6. QTTimeCode’s Timecode Options dialog box.

As you can see, QTTimeCode allows the user to specify the name of the media source (which might be the name of the tape from which the video was digitized). The user can also indicate whether the timecode should be displayed at all, and (if so) whether the timecode should be displayed below the video. QTTimeCode allows the user to specify the font in which the timecode is displayed and the starting time of the timecodes.

It’s worth emphasizing that a timecode track in a QuickTime movie does not affect the actual timing of the movie playback. QuickTime’s own timing facilities always determine the rate at which frames in a movie are displayed. If, for example, we play a movie back at twice its normal rate, the timecodes would also zip by at twice the speed of a wall clock.

TIMECODE TRACKS

We add a timecode track to a movie in exactly the same way we’ve added other kinds of tracks. The main difference, of course, is that the media type of the new track will be `TimeCodeMediaType`. To know how to add a timecode track to an existing movie, we need to know only two things: (1) the format of the data in a timecode media sample; and (2) the structure of a sample description for a timecode media sample. Let’s tackle

these issues in reverse order. We'll also consider how to set the placement of the timecode track and configure some of the text options.

Creating a Timecode Sample Description

A *timecode sample description* is defined by the TimeCodeDescription data type, like this:

```
struct TimeCodeDescription {
    long      descSize;
    long      dataFormat;
    long      resvd1;
    short     resvd2;
    short     dataRefIndex;
    long      flags;
    TimeCodeDef timeCodeDef;
    long      srcRef[1];
};
```

The first five fields are common to all sample descriptions, and we need to fill in only the descSize and dataFormat fields:

```
TimeCodeDescriptionHandle myDesc = NULL;
long      mySize;

mySize = sizeof(TimeCodeDescription);
myDesc =
    (TimeCodeDescriptionHandle)NewHandleClear(mySize);
if (myDesc == NULL)
    goto bail;

(**myDesc).descSize = mySize;
(**myDesc).dataFormat = TimeCodeMediaType;
```

The flags field is reserved and should be set to 0. The timeCodeDef field is a *timecode definition structure* that contains information about the desired format of the timecode. A timecode definition structure is defined by the TimeCodeDef structure:

```
struct TimeCodeDef {
    long      flags;
    TimeScale fTimeScale;
    TimeValue frameDuration;
    UInt8     numFrames;
    UInt8     padding;
};
typedef struct TimeCodeDef TimeCodeDef;
```

The flags field of the TimeCodeDef structure contains a set of flags that specify information about the format of the timecode. Currently, these flags are defined:

```
enum {
    tcDropFrame      = 1 << 0,
    tc24HourMax      = 1 << 1,
    tcNegTimesOK      = 1 << 2,
    tcCounter         = 1 << 3
};
```

The tcDropFrame flag, of course, indicates whether the timecode is drop-frame or non-drop-frame timecode. The tc24HourMax flag indicates whether the hours digits return to 0 at 24 hours or keep advancing. The tcNegTimesOK flag indicates whether negative timecode values are allowed. Finally, the tcCounter flag indicates whether to display the simple counter instead of a SMPTE timecode.

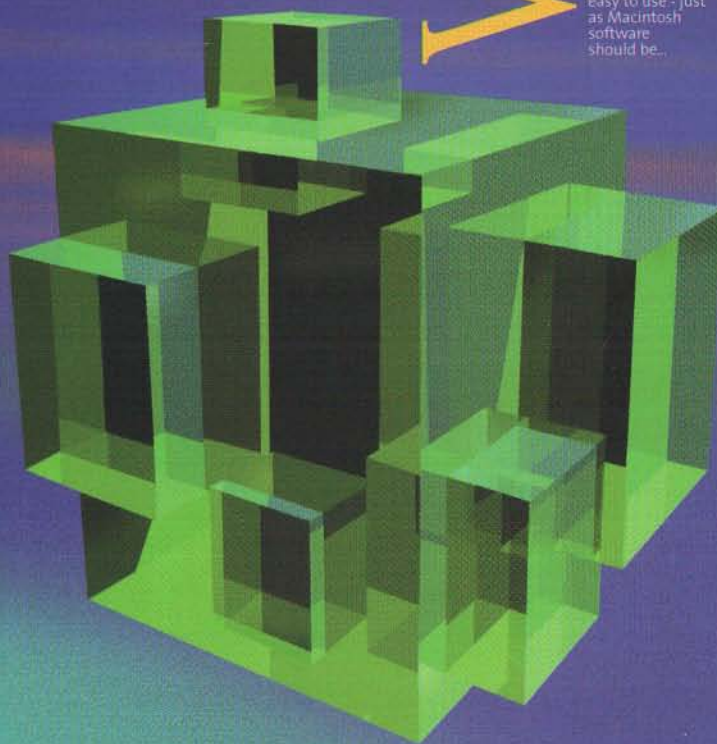
Come and see
our OS X version at
MacWorld Expo
San Francisco

The Version
Control Tool for
CodeWarriors

VOODOO Server



The successor of
our award-winning
VOODOO. More
powerful and still
easy to use - just
as Macintosh
software
should be...



www.unisoftwareplus.com
voodoo@unisoftwareplus.com

Available
at
DEPOT
877-DEPOT-NOV or 800-494-9797
to 800-494-9798
<http://www.depot.com>
order@depot.com

The numFrames field of the TimeCodeDef structure indicates the number of frames in each second of media data in the target track. If the simple counter is being displayed, this field indicates the number of frames that make up each increment of the counter. The frameDuration field indicates the duration, in time units defined by the fTimeScale field, of each frame in the target media.

QTTimeCode keeps track of most of these settings by defining a set of global variables (which makes it easy to get information into and out of the Timecode Options dialog box). Using some of these globals, we set up the timecode definition structure as shown in Listing 2.

Listing 2: Setting up a timecode definition structure

QTTC_AddTimeCodeToMovie

```
// set the timecode format information flags
if (gUseTimeCode) {
    myFlags = 0L;
    if (gDropFrameVal)
        myFlags |= tcDropFrame;
    if (gIsNeg)
        myFlags |= tcNegTimesOK;
    if (g24Hour)
        myFlags |= tc24HourMax;
} else {
    myFlags = tcCounter;
}

myTCDef.flags = myFlags;
myTCDef.fTimeScale = gTimeScale;
myTCDef.frameDuration = gFrameDur;
myTCDef.numFrames = gNumFrames;
```

The last field in a timecode sample description is the srcRef field, which is defined as an array containing a single long integer. This field contains information about the source of the timecode values. As mentioned earlier, this is typically the name of the tape or other source of the media data from which the timecode values were originally captured. The interesting thing here is that the data in this field must be formatted in the same way as a piece of movie or track user data. As a result, we can create the source information using QuickTime's user data functions like NewUserData and AddUserDataText. But the source information is not stored in the same location as movie or track user data; instead, it's stored in the timecode sample description itself. The timecode media handler provides a function, TCSetSourceRef, that we can use to put the source information into the timecode sample description. Listing 3 shows how we set the source information for a timecode track. Notice that the user data item is of type TCSOURCEREFNAMEType.

Listing 3: Setting the source information

QTTC_AddTimeCodeToMovie

```
UserData    myUserData;

myErr = NewUserData(&myUserData);
if (myErr == noErr) {
```

```
    Handle    myNameHandle = NULL;

    myErr = PtrToHand(&gSrcName[1], &myNameHandle,
                      gSrcName[0]);

    if (myErr == noErr) {
        myErr = AddUserDataText(myUserData, myNameHandle,
                                TCSOURCEREFNAMEType, 1, langEnglish);
        if (myErr == noErr)
            TCSetSourceRef(myHandler, myDesc, myUserData);
    }

    if (myNameHandle != NULL)
        DisposeHandle(myNameHandle);

    DisposeUserData(myUserData);
}
```

Why isn't the source information stored as part of the track's user data (in an atom of type 'udta')? Part of the reason is that there can be more than one timecode sample in a timecode track, and each sample may be associated with timecode information from a different source. So it makes sense to attach the source information to the timecode sample description. In any event, we really don't need to worry about where the source information is stored, since we shall always use the functions TCSetSourceRef and TCGetSourceRef to set and get that information. (For what it's worth, QTTimeCode will create only one timecode sample, which spans length of the entire target track.) TCSetSourceRef updates the descSize field of the timecode sample description, so we don't need to do this ourselves.

Creating a Timecode Media Sample

Now we need to see how to create a media sample for our timecode track. The sample description contains information about the format and source of the timecode. The start time and duration of the media sample indicate the start time and duration of the timecode display. All that remains is to specify the timecode value of the first frame of the target track segment. That is to say, when the timecode media handler encounters a timecode sample, it needs to know what the timecode value is at that moment. This is precisely what's contained in a timecode media sample.

A timecode media sample contains a long integer that specifies the timecode value of the first frame in the target track that is to have a timecode value associated with it. The timecode value is stored as a frame number, which is converted automatically to a timecode value by the timecode media handler. The timecode media handler provides functions for converting timecode values into frame values and *vice versa*. To see how this works, let's suppose that we want the first frame of a movie to have the timecode 08:23:11;00. Suppose further that the four parts of this timecode value are contained in the global variables gHours, gMinutes, gSeconds, and gFrames. Ultimately we want to call the function TCTimeCodeToFrameNumber to determine the frame number that we'll put into the media sample. We

specify the timecode value to TCTimeCodeToFrameNumber by passing it a pointer to a *time code record*, defined like this:

```
union TimeCodeRecord {
    TimeCodeTime    t;
    TimeCodeCounter  c;
};
typedef union TimeCodeRecord TimeCodeRecord;
```

This is a union that contains either a *timecode time record* or a *timecode counter record*:

```
struct TimeCodeTime {
    UInt8    hours;
    UInt8    minutes;
    UInt8    seconds;
    UInt8    frames;
};
typedef struct TimeCodeTime TimeCodeTime;

struct TimeCodeCounter {
    long    counter;
};
typedef struct TimeCodeCounter TimeCodeCounter;
```

Since we've got a timecode value that we want to convert into a frame number, we'll use the t variant, like this:

```
myTCTRec.t.hours = (UInt8)gHours;
myTCTRec.t.minutes = (UInt8)gMinutes;
myTCTRec.t.seconds = (UInt8)gSeconds;
myTCTRec.t.frames = (UInt8)gFrames;
if (gIsNeg)
    myTCTRec.t.minutes |= tctNegFlag;
```

Notice that the minutes field of the time code time record should have the bit tctNegFlag (namely, 0x80) set if the timecode value is to be interpreted as a negative value.

We're finally ready to create the timecode media sample data. Since AddMediaSample wants a handle to the media data, we need to allocate a relocatable block large enough to hold a long integer and then call TCTimeCodeToFrameNumber to convert the timecode stored in myTCTRec into a frame number. Then we need to ensure that the frame number is stored in big-endian format, like this:

```
myFrameHandle = (long **)NewHandle(sizeof(long));
if (myFrameHandle == NULL)
    goto bail;

myErr = TCTimeCodeToFrameNumber(myHandler,
    &(**myDesc).timeCodeDef, &myTCTRec,
    *myFrameHandle);

**myFrameHandle = EndianS32_NtoB(**myFrameHandle);
```

Then we proceed as usual, calling BeginMediaEdits, AddMediaSample, EndMediaEdits, and InsertMediaIntoTrack.

Setting the Timecode Track Geometry

For simplicity, the QTTimeCode application uses the first video track in a movie as the target track (that is, the track that contains a track reference to the timecode track). We'll call GetTrackDimensions on the target track to get the desired width of the timecode track. For the initial

FUNNELWEB
VERSION 4



Better, Stronger, Faster

Intelligent Web site Monitoring and Analysis Software

Speed, intuitive user interface, accuracy and in-depth analysis have always made Funnel Web the intelligent choice for Web site analysis.

Now includes pdf output, incremental analysis, cluster analysis and streaming media reports.

Better, Stronger, Faster

height of the timecode track, we'll use a predefined constant. Ideally, however, the height of the timecode track should depend on the height of the timecode display, which depends of course on the font and point size of the characters in the timecode display. We can retrieve information about the timecode display by calling `TCGetDisplayOptions`, which returns information in a *text options structure*, defined by the `TCTextOptions` data type:

```
struct TCTextOptions {
    short      txFont;
    short      txFace;
    short      txSize;
    short      pad;
    RGBColor   foreColor;
    RGBColor   backColor;
};
typedef struct TCTextOptions  TCTextOptions;
```

The `txFont` field contains the font number of the timecode display font. When setting up a timecode track, we want to do two things with this font. First, we want to reset the `txFont` field to the font number of the font that the user selected in our application's Timecode Options dialog box (see Figure 6 again). Then we want to measure that font to determine the maximum height for our timecode track.

When the user closes the Timecode Options dialog box by clicking the OK button, `QTTimeCode` gets the index and name of the currently-selected font by executing these two lines of code:

```
gFontIndex = GetControlValue(myControl);
GetMenuItemText(myMenu, gFontIndex, gFontName);
```

Then, when creating a timecode track, `QTTimeCode` uses the font name stored in the `gFontName` global variable to find the appropriate font number:

```
TCGetDisplayOptions(myHandler, &myTextOptions);
GetFNum(gFontName, &myTextOptions.txFont);
TCSetDisplayOptions(myHandler, &myTextOptions);
```

Now we need to determine the maximum height of the timecode display. We can do this by getting a representative string containing a timecode and then measuring that string in the current font, style, and size:

```
// use the starting time to figure out the dimensions of track
TCTimeCodeToString(myHandler,
                    &myTCDef, &myTCRec, myString);
TextFont(myTextOptions.txFont);
TextFace(myTextOptions.txFace);
TextSize(myTextOptions.txSize);
GetFontInfo(&myFontInfo);

// calculate track width and height based on text
myTCHeight = FixRatio(myFontInfo.ascent +
                      myFontInfo.descent + 2, 1);
```

You'll notice that we determine the height of the timecode track by adding the distance from the font baseline to the ascent line and the distance from the baseline to the

descent line (plus a couple of pixels of extra space). In some fonts, the numerals have no descenders, so this calculation will tend to make the timecode look out of center vertically within the timecode track. Devising a better algorithm is left as an exercise for the reader.

Now we've determined the proper height and width for the timecode track. At this point, we can reset the track dimensions:

```
SetTrackDimensions(myTrack, myWidth, myTCHeight);
```

Then, if the user has indicated that the timecode track should appear below the video track, we can reset the track matrix, like this:

```
GetTrackMatrix(myTrack, &myMatrix);
if (gDisplayBelowVideo)
    TranslateMatrix(&myMatrix, 0, myHeight);
SetTrackMatrix(myTrack, &myMatrix);
```

The timecode track is now positioned at the desired location with the appropriate width and height.

We can configure the track to be displayed by calling `TCSetTimeCodeFlags` with the `tcdfShowTimeCode` flag.

```
TCSetTimeCodeFlags(myHandler,
                    gDisplayTimeCode ? tcdfShowTimeCode : 0,
                    tcdfShowTimeCode);
```

A timecode track must be enabled in order for it to be displayed, so we should call `SetTrackEnabled` before calling `TCSetTimeCodeFlags`.

Creating a Track Reference

The final thing we need to do, when creating a timecode track, is to create a track reference from the target track to the timecode track, like this:

```
myErr = AddTrackReference(myTypeTrack, myTrack,
                          TimeCodeMediaType, NULL);
```

Listing 4 shows our complete function for creating a timecode track in a movie.

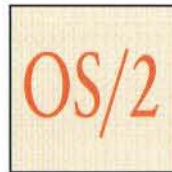
Listing 4: Adding a timecode track to a QuickTime movie

```
QTTC_AddTimeCodeToMovie
OSErr QTTC_AddTimeCodeToMovie
    (Movie theMovie, OSType theType)
{
    Track          myTypeTrack = NULL;
    Track          myTrack = NULL;
    Media          myMedia = NULL;
    MediaHandler   myHandler = NULL;
    TimeCodeDef    myTCDef;
    TimeCodeRecord myTCRec;
    Str63          myString;
    TimeValue      myDuration;
    MatrixRecord   myMatrix;
    Fixed          myWidth;
    Fixed          myHeight;
    Fixed          myTCHeight;
    long           myFlags = 0L;
    TCTextOptions  myTextOptions;
    FontInfo       myFontInfo;
```

The *True* BASIC[®] story
is a *powerful* ~~simple~~ story.

Write your code once.

Run it (without change)



here, here, here, here, here, here & here!

Demos and information from our web site:

<http://www.truebasic.com>

TRUE BASIC INC • Founded by the *Inventors* of BASIC

PO BOX 5428 • WEST LEBANON NH 03784-5428 • 800 436-2111 • 802 296-2711

```

TimeCodeDescriptionHandle myDesc = NULL;
long **myFrameHandle;
OSErr myErr = noErr;

// get the (first) track of the specified type; this track determines the width of
the new timecode track
myTypeTrack = GetMovieIndTrackType(theMovie, 1,
                                   theType, movieTrackMediaType);
if (myTypeTrack == NULL) {
    myErr = trackNotInMovie;
    goto bail;
}

// get the dimensions of the target track
GetTrackDimensions(myTypeTrack, &myWidth, &myHeight);

// create the timecode track and media
myTrack = NewMovieTrack(theMovie, myWidth,
                        kTimeCodeTrackSize, kNoVolume);
if (myTrack == NULL)
    goto bail;

myMedia = NewTrackMedia(myTrack, TimeCodeMediaType,
                        GetMovieTimeScale(theMovie), NULL,
0);
if (myMedia == NULL)
    goto bail;

myHandler = GetMediaHandler(myMedia);
if (myHandler == NULL)
    goto bail;

// fill in a timecode definition structure; this becomes part of the timecode
description

// set the timecode format information flags
if (gUseTimeCode) {
    myFlags = 0L;
    if (gDropFrameVal)
        myFlags |= tcDropFrame;
    if (gIsNeg)
        myFlags |= tcNegTimesOK;
    if (g24Hour)
        myFlags |= tc24HourMax;
} else {
    myFlags = tcCounter;
}

myTCDef.flags = myFlags;
myTCDef.fTimeScale = gTimeScale;
myTCDef.frameDuration = gFrameDur;
myTCDef.numFrames = gNumFrames;

// fill in a timecode record
if (gUseTimeCode) {
    myTCRec.t.hours = (UInt8)gHours;
    myTCRec.t.minutes = (UInt8)gMinutes;
    myTCRec.t.seconds = (UInt8)gSeconds;
    myTCRec.t.frames = (UInt8)gFrames;
    if (gIsNeg)
        myTCRec.t.minutes |= tcNegFlag;
} else {
    myTCRec.c.counter = gCounterVal;
}

// figure out the timecode track geometry

// get display options to calculate box height
TCGetDisplayOptions(myHandler, &myTextOptions);
GetFNum(gFontName, &myTextOptions.txFont);
TCSetDisplayOptions(myHandler, &myTextOptions);

// use the starting time to figure out the dimensions of track
TCTimeCodeToString(myHandler, &myTCDef, &myTCRec,
                    myString);
TextFont(myTextOptions.txFont);
TextFace(myTextOptions.txFace);
TextSize(myTextOptions.txSize);

```

```

GetFontInfo(&myFontInfo);

// calculate track width and height based on text
myTCHeight = FixRatio(myFontInfo.ascent +
                      myFontInfo.descent + 2, 1);
SetTrackDimensions(myTrack, myWidth, myTCHeight);

GetTrackMatrix(myTrack, &myMatrix);
if (gDisplayBelowVideo)
    TranslateMatrix(&myMatrix, 0, myHeight);

SetTrackMatrix(myTrack, &myMatrix);
SetTrackEnabled(myTrack, gDisplayTimeCode ? true :
false);

TCSetTimeCodeFlags(myHandler, gDisplayTimeCode ?
                    tcdfShowTimeCode : 0, tcdfShowTimeCode);

// edit the track media
myErr = BeginMediaEdits(myMedia);
if (myErr == noErr) {
    long mySize;
    UserData myUserData;

// create and configure a new timecode description handle
mySize = sizeof(TimeCodeDescription);
myDesc = (TimeCodeDescriptionHandle)
NewHandleClear(mySize);

if (myDesc == NULL)
    goto bail;

(**myDesc).descSize = mySize;
(**myDesc).dataFormat = TimeCodeMediaType;
(**myDesc).timeCodeDef = myTCDef;

// set the source identification information

// the source identification information for a timecode track is stored
// in a user data item of type TCSrcRefNameType
myErr = NewUserData(&myUserData);
if (myErr == noErr) {
    Handle myNameHandle = NULL;

    myErr = PtrToHand(&gSrcName[1], &myNameHandle,
                     gSrcName[0]);

    if (myErr == noErr) {
        myErr = AddUserDataText(myUserData,
                                myNameHandle,
                                TCSrcRefNameType, 1,
                                langEnglish);
        if (myErr == noErr)
            TCSetSourceRef(myHandler, myDesc, myUserData);
    }

    if (myNameHandle != NULL)
        DisposeHandle(myNameHandle);

    DisposeUserData(myUserData);
}

// add a sample to the timecode track
myFrameHandle = (long **)NewHandle(sizeof(long));
if (myFrameHandle == NULL)
    goto bail;

myErr = TCTimeCodeToFrameNumber(myHandler,
                                &(**myDesc).timeCodeDef, &myTCRec,
                                *myFrameHandle);

// the data in the timecode track must be big-endian
**myFrameHandle = EndianS32_NtoB(**myFrameHandle);

myDuration = GetMovieDuration(theMovie);
// since we created the track with the same timescale as the movie,
// we don't need to convert the duration

myErr = AddMediaSample(myMedia,
                       (Handle)myFrameHandle,
                       0,

GetHandleSize((Handle)myFrameHandle),
myDuration,
(SampleDescriptionHandle)myDesc,
1, 0, 0);
if (myErr != noErr)

```

```

    goto bail;
}

myErr = EndMediaEdits(myMedia);
if (myErr != noErr)
    goto bail;

myErr = InsertMediaIntoTrack(myTrack, 0, 0,
                             myDuration, fixed1);
if (myErr != noErr)
    goto bail;

// create a track reference from the target track to the timecode track
myErr = AddTrackReference(myTypeTrack, myTrack,
                          TimeCodeMediaType, NULL);

bail:
if (myDesc != NULL)
    DisposeHandle((Handle)myDesc);

if (myFrameHandle != NULL)
    DisposeHandle((Handle)myFrameHandle);

return(myErr);
}

```

GETTING INFORMATION ABOUT A TIMECODE TRACK

The timecode media handler provides several functions that we can use to get and set information about a timecode track in a movie. For example, if we want to get the current timecode value for a movie, we can call the `TCGetCurrentTimeCode` function. Listing 5 defines the `QTTC_ShowCurrentTimeCode` function, which `QTTimeCode` uses to get the current timecode value and display it in a dialog box to the user.

Listing 5: Displaying the current timecode value

QTTC_ShowCurrentTimeCode

```

void QTTC_ShowCurrentTimeCode (Movie theMovie)
{
    MediaHandler      myHandler = NULL;
    HandlerError       myErr = noErr;
    TimeCodeDef        myTCDef;
    TimeCodeRecord     myTCRec;

    myHandler = QTTC_GetTimeCodeMediaHandler(theMovie);
    if (myHandler != NULL) {

        // get the timecode for the current movie time
        myErr = TCGetCurrentTimeCode(myHandler, NULL, &myTCDef,
                                     &myTCRec, NULL);

        if (myErr == noErr) {
            Str255 myString;

            myErr = TCTimeCodeToString(myHandler, &myTCDef,
                                     &myTCRec, myString);

            if (myErr == noErr)
                QTTC_ShowStringToUser(myString);
        }
    }
}

```

As you can see, `QTTC_ShowCurrentTimeCode` first retrieves the instance of the timecode media handler and then calls `TCGetCurrentTimeCode` to get the current timecode value. Then it passes the timecode definition structure and timecode record filled in by `TCGetCurrentTimeCode` to the `TCTimeCodeToString` function, to convert the timecode

Always Thinking

Professional Macintosh & Internet Development

Always Thinking's professional developers will help you meet your Macintosh and Internet deadlines! So if you're...

- on a tight deadline and need additional talent
- losing valuable development time debugging
- having trouble finding good developers

... **Always Thinking's** team of experienced programmers will provide you with a timely and affordable solution.

We deliver more than code — a complete project. Our software engineers work with you to:

- Create clear, solid project specifications
- Design and develop your application or web site
- Tune and optimize your software's performance
- Thoroughly test your application or site
- Completely document your project
- Provide training to your team

Commercial Product Development

Do you have an exciting idea for an application? Turn to **Always Thinking** to make it a reality. We have firsthand experience developing and shipping award-winning commercial applications for our clients and our own Thinking Home, a 2000 Apple Design Award winner.

Web Site Design & Development

Get a sound e-commerce system tailored for both your immediate needs and long-term growth. Our engineers can develop the Internet applications to transform your company into an e-business.

Successful web sites are more than graphics and code. We have the Internet marketing know-how to ensure your site is an effective business tool.

Realize substantial savings by moving to online pre-sales information, ordering and support.

Tell us about your project, toll-free

(800) 252-6479

(703) 478-0181 x103

Always Thinking, Inc.
27 James Byrnes Street
Beaufort, SC 29902

www.alwaysthinking.com sales@alwaysthinking.com



value into a string. Finally, QTTC_ShowCurrentTimeCode calls the application function QTTC_ShowStringToUser to show the string to the user.

Similarly, we can display a timecode track's source information to the user by calling the QTTC_ShowTimeCodeSource function defined in Listing 6.

Listing 6: Displaying the timecode source information

```

QTTC_ShowTimeCodeSource

void QTTC_ShowTimeCodeSource (Movie theMovie)
{
    MediaHandler    myHandler = NULL;
    HandlerError    myErr = noErr;
    UserData        myUserData;

    myHandler = QTTC_GetTimeCodeMediaHandler(theMovie);
    if (myHandler != NULL) {

        // get the timecode source for the current movie time
        myErr = TCGetCurrentTimeCode(myHandler, NULL, NULL,
                                     NULL, &myUserData);

        if (myErr == noErr) {
            Str255    myString = " [No source name!]";
            Handle    myNameHandle = NewHandleClear(0);

            GetUserDataText(myUserData, myNameHandle,
                           TCSrcRefNameType, 1, langEnglish);
            if (GetHandleSize(myNameHandle) > 0) {
                BlockMove(*myNameHandle, &myString[1],
                          GetHandleSize(myNameHandle));
                myString[0] = GetHandleSize(myNameHandle);
            }

            if (myNameHandle != NULL)
                DisposeHandle(myNameHandle);

            QTTC_ShowStringToUser(myString);

            DisposeUserData(myUserData);
        }
    }
}

```

One thing to notice here is that the TCGetCurrentTimeCode function returns, through its last parameter, a user data list that contains the source information of a timecode track. (We didn't need the source information in Listing 5, so there we passed NULL for the last parameter.) Once we've successfully retrieved that information, we can use the Movie Toolbox user data function GetUserDataText to find the source information and then display it to the user.

SHOWING AND HIDING A TIMECODE TRACK

As we've seen in earlier articles, we can show or hide a specific track in a movie by enabling and disabling that track using the Movie Toolbox function SetTrackEnabled. For example, we can toggle the visible state of a track with this line of code:

```
SetTrackEnabled(myTrack, !GetTrackEnabled(myTrack));
```

With a timecode track, we also want to keep the media handler informed of the visible state of the track by

setting the timecode flags appropriately, like this:

```

TCGetTimeCodeFlags(myHandler, &myFlags);
myFlags ^= tcdfShowTimeCode;
TCSetTimeCodeFlags(myHandler, myFlags,
tcdfShowTimeCode);

```

Here, we simply retrieve the current timecode flags, toggle the tcdfShowTimeCode bit, and then send the new set of flags back to the timecode media handler. The third parameter to the TCSetTimeCodeFlags function is a mask that indicates which bits in the myFlags parameter to consider. As you can see, we want the media handler to change only the tcdfShowTimeCode bit itself. Listing 7 shows the complete definition of the QTTC_ToggleTimeCodeDisplay function.

Listing 7: Toggling the visible state of a timecode track

```

QTTC_ToggleTimeCodeDisplay

void QTTC_ToggleTimeCodeDisplay (MovieController theMC)
{
    Movie        myMovie = MCGetMovie(theMC);
    Track        myTrack = NULL;
    MediaHandler myHandler = NULL;
    long         myFlags = 0L;

    // get the (first) timecode track in the specified movie
    myTrack = GetMovieIndTrackType(myMovie, 1,
                                   TimeCodeMediaType, movieTrackMediaType);
    if (myTrack != NULL) {

        // get the timecode track's media handler
        myHandler = QTTC_GetTimeCodeMediaHandler(myMovie);
        if (myHandler != NULL) {

            // toggle the show-timecode flag
            TCGetTimeCodeFlags(myHandler, &myFlags);
            myFlags ^= tcdfShowTimeCode;
            TCSetTimeCodeFlags(myHandler, myFlags,
                               tcdfShowTimeCode);

            // toggle the track enabled state
            SetTrackEnabled(myTrack, !GetTrackEnabled(myTrack));

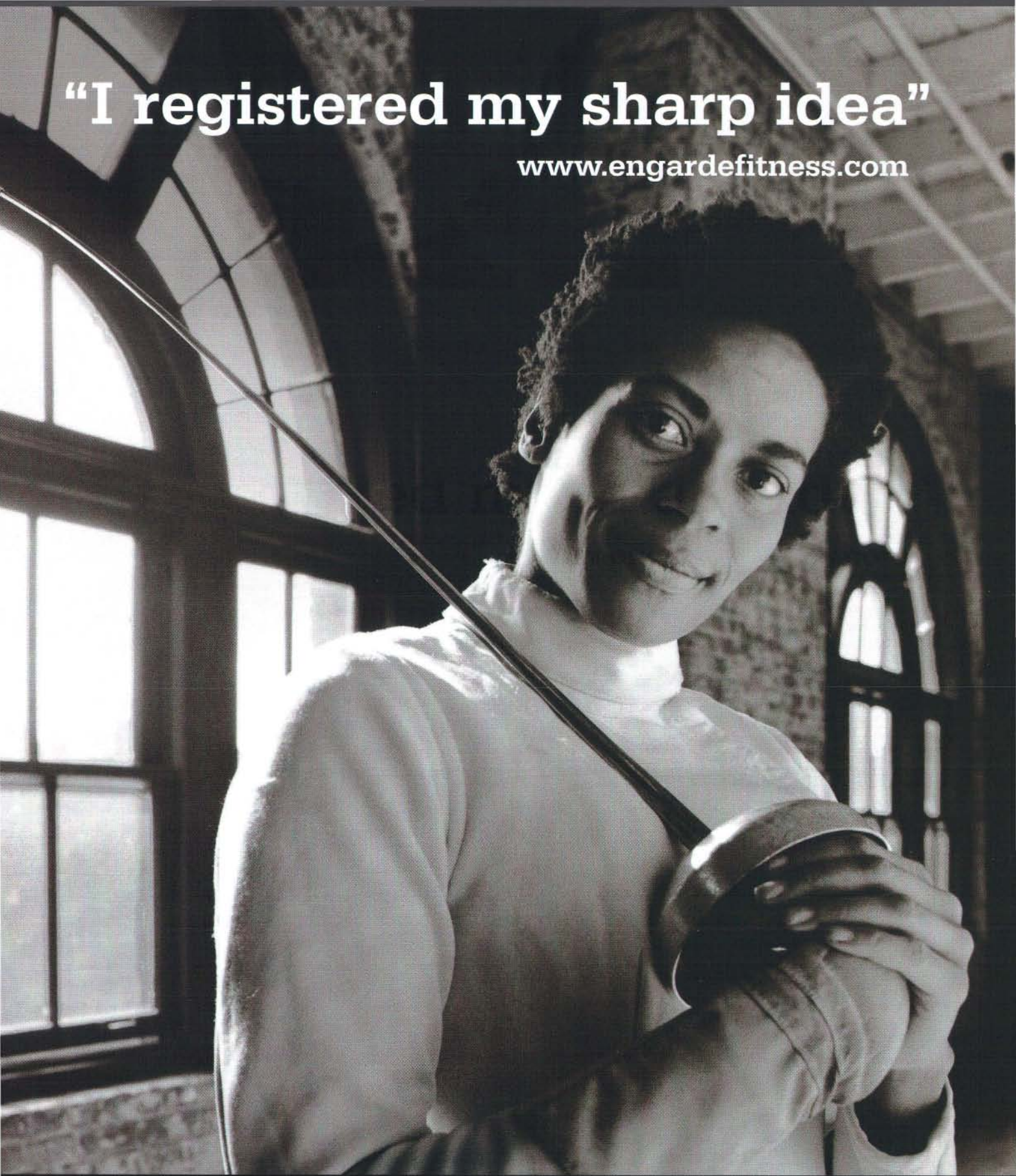
            // now tell the movie controller the movie has changed,
            // so that the movie rectangle gets updated correctly
            MCMovieChanged(theMC, myMovie);
        }
    }
}

```

Notice that we call MCMovieChanged to prompt the movie controller to reset its state based on the current properties of the associated movie. This causes the movie controller to recalculate the controller rectangle and redraw the movie.

CONCLUSION

As we've seen, it's quite easy to create and manipulate timecode tracks. These tracks take up very little space in a movie file, and they can be extremely useful in coordinating QuickTime movies with their external source material. While timecode display is principally useful to video or audio editors, the counter display can serve a variety of purposes.



"I registered my sharp idea"

www.engardefitness.com

Sharon Monplaisir, Olympic fencer and entrepreneur, registered her domain name to put her business online at register.com. Visit us at www.register.com or call us at 1-800-7-WWW-NET, and we'll help you register your domain name right over the phone.

register
● **com**TM
the *first* step on the webSM

by Xplain Staff

KICK-OFF!

Kick-Off! 1.0 is the football-shaped USB cousin of Rebound! 2.0, the remote server godsend produced by Sophisticated Circuits. Because of subtle differences between the USB and ADB communication channels, Kick-Off! requires a slightly more complicated set up than Rebound. Rebound is a simple ADB device that plugs into your computer's ADB port and allows your keyboard to plug into it, thereby intercepting signals between the two. Kick-Off! has a slightly more complicated setup. It plugs into your computer's power outlet, and has your existing power cord plug into it, while at the same time plugging its USB cable into your computer's spare USB port. Once you've got the hardware set up, just run the installer and you can say goodbye to remote server downtime...almost.

Kick-Off!'s most basic and helpful feature is the ability to restart the computer when the system crashes, but there are some qualifications that should be made. The main issue I had with Kick-Off! was not that it failed to detect crashes and restart properly, but that it did it too often, sometimes when the machine had not even crashed! Beware of applications that hog the system's processing abilities. For example, in my first test of Kick-Off!, I wanted to take the processor out for a little spin, so I started up Quake III and began blowing bots to oblivion. Although I had almost instantly forgotten about my Kick-Off! test, I was cruelly reminded when the machine abruptly rebooted exactly one minute into my game (I had set the crash detection timer to one minute)! However, when I restarted Quake and changed from full-screen mode to a window, it worked fine. Now, nobody in their right mind would ever play Quake in a window, so if you are running applications like this, Kick-Off! will not make you happy.

Still, most servers aren't running applications like Quake III, and it would be unfair of me to base my entire review upon this one aspect of Kick-Off!. In most other situations, Kick-Off! performed exactly as planned. Its reboot mechanism works well when the system does in fact crash, and can be used in conjunction with application crashes as well, which is a particularly helpful feature if you are using Kick-Off! on a remote server. Kick-Off! is able to respond to application crashes in a variety of different ways; it can restart the application, or the computer, or both, or any combination of either. I used Crash Test, I freeware application (also from Sophisticated Circuits) that causes several different types of system and application crashes, to test Kick-Off!'s effectiveness. For two types of system crashes, Kick-Off! detected the crashes admirably and restarted the computer as planned. Crash Test can also perform two types of application crashes: it can hang an application, and it can allow its application timer to expire. The latter function is designed specifically for Sophisticated Circuits' equipment, and not surprisingly, Kick-Off! was able to accurately detect this type of crash and relaunch the application. However, when I told Crash Test to

"hang" itself, Kick-Off! understood this as a system crash and performed a hard restart when the system timer expired. So, while Kick-Off!'s system crash detection works quite well, its application crash detection is not quite as effective.

I recommend using Kick-Off! in place of Rebound on remote Mac servers that support USB (rather than ADB) devices. However, it is somewhat limited in its ability to correctly ascertain whether a particular application has crashed or not, and you will not find it reliable without an additional line of defense. Make sure that if you are using Kick-Off! to relaunch applications, you have adequately tested them and are sure that they support the Kick-Off! application timer. While Kick-Off! is not the savior of network administrators, it will help to reduce those 3 AM trips to the Office for server restarting.

SYSLOGD

If you're like me, you want to know as much as possible about what's happening with your network at all times. If you're having unidentified problems and you want to see what your switches are doing, you can be in a tough spot if you're using a Mac. In steps Syslogd, a shareware utility program written by Brian Bergstrand. The current version of Syslogd (2.1.8) allows an administrator to view all sorts of actions taking place on his/her network.

Syslogd is, of course, titled after its UNIX namesake. Accordingly, it brings you right down to the nitty-gritty computer instructions that the Mac OS historically prevented the user from seeing. However, for the network administrator interested in such matters, Syslogd serves up a hot plate of delicious, abstruse UNIX-style information. Easily downloaded at Bergstrand's web site www.classicalguitar.net/brian/apps/syslogd, the application features a quick AppleScript installer to get you started. (You will need AppleScript installed to perform some of Syslogd's convenient functions, but the actual application will run without it). Make it a point to also download Stairways' Interarchy 3.8, which provides you with some helpful tools (including a UDP Ping-er) if you get stuck using Syslogd. The Syslogd application should load and begin running on system start-up, but you'll want to run the Syslog Console application as well, at least at first. In order to get information displayed in the console window, I had to set up a CONSOLE log in the Syslogd Admin control panel, which is a piece of cake. After a Syslogd restart, you should start seeing messages zip across the console window. If you choose not to use the console application, you can still view the logs in your favorite text-editor, after running the Cycle Logs AppleScript included in the package. Presumably, you should be able to find the information you're looking for somewhere in the Syslogd logs.

Unfortunately, Syslogd 2.1.8 will not run on your old LCIII, or any other 68k machine. In addition, Syslogd is not compatible with Timbuktu, and you can expect a system crash if ever the twain shall

meet. While setting up the application and getting messages to appear in the console window was not particularly intuitive (especially if Interarchy is not available), Bergstrand himself is a fantastic resource if you get into trouble.

LINK TESTER 2.5

Recently, I had the opportunity to evaluate Link Tester 2.5 from VSE Software on our www.mactech.com website. Link Tester is a program that crawls its way through your web site following each link in your site one-by-one (it can test web pages, images, downloadable file links, and even JavaScript links) and reporting any pages that have problems displaying or any pages that link to dead material on the World Wide Web.

The program has a rather pleasant, if minimalistic, user interface and is pretty straightforward to use. You simply install the program on your machine, enter the web address of your web site, and let Link Tester do all the work. You may tell Link Tester to check just links on pages you specify, or to follow all links on your site recursively. The default mode is to check all links on your site recursively.

As a link is broken or otherwise malfunctioning it is added to a HTML-formatted report that is presented to you at the end of the site evaluating process. This report lists all links that were checked, the broken ones, and the reason(s) for failure. There are options to save this report and you can even click on the link of the bad page just to see that indeed the link is really broke. In addition you can also click on an error message for a more detailed description of that message.

Link Tester will attempt to open multiple connections to the website being tested so that it can complete its job faster. You are allowed some control over this by being able to choose a 'Server Load' level of 5 different steps between Very High and Very Low. It is recommended that when connecting to remote servers you should always use the Very Low setting out of respect to those server operators.

MACSENSE CT-2? CABLE TESTER

For most of us, working with computer cables is about as enjoyable as being a White House intern. You have to make sense of indistinguishable strands of gray matter, often woven together in an intricate pattern of loops, tangles, and knots. In order to get at them, you are often required to move heavy furniture, or wedge yourself in unsavory places that even the vacuum cleaner has forsaken. All too often, all of your energy goes for naught when you later discover that you put a bad cable in place. For anyone who knows this frustration, the MacSense Cable Tester is for you.

The CT-2 is a simple, yet ingeniously designed device that will save you hours of troubleshooting agony. It has six main ports: two each of RJ-45 (Ethernet), RJ-11 (telephone), and MiniDin (serial). Plug each end of a cable into a port, press a button, and the CT-2 will tell you which pins or wires within the cable are responding, and how well. It can also tell you whether the connection is parallel or non-parallel. Never again will you confuse a crossover cable and a patch cable, nor will you waste time installing a cable that doesn't work, or doesn't work well. For older networks, the CT-2 has a BNC port that can accommodate a 10base2 T-connector and determine its terminator value.

Mac OS X

Porting & Development Showcase

Mac OS X

Making the Move

With **mac OS X** just over the **HORIZON**,
Mac **developers** everywhere have a
major need for **CARBON** and **cocoa**
application porting, **AQUA** user interface
implementation, and **DRIVER** development
services. Toward that end, several
high-quality **SOFTWARE** engineering
firms, in association with the **APPLE**
DEVELOPER CONNECTION, are offering these
services at very **attractive** discounts
to all **ADC** Select and Premier members.

The following pages
are those vendors that
are part of the Mac OS X

Porting & Development Showcase.

Get the power and experience you need from

PROSOFT

e n g i n e e r i n g , i n c .

With over thirteen years in providing programming service to the Mac community. Prosoft is committed in delivering the ultimate quality software products and service.

Got
OS X?

Our Engineers have extensive knowledge in:

- Macintosh PPC Drivers, Shims and Extensions
- Macintosh Power Plant Applications, Mac OS X applications and drivers
- Carbon application porting for Mac OS X
- Multimedia and QuickTime Applications
- Unix/Linux Console Applications, including Unix Solaris, xBSD Drivers, and Linux Drivers

Other areas of expertise

- Windows 95, 98
- Window CE
- Windows NT, 2000
- Unix / Linux
- Java
- Embedded OS
- RDBMS
- Network Consulting

SOME THINGS ARE
BEST LEFT TO EXPERTS



Don't want to miss the mark while moving to **Mac OS X**. Come to RecoSoft. We're the experts in developing software for the Macintosh platform. Our quiver of skills include Macintosh Application Design & Development, Cross Platform (Mac/Win/Linux) Development Services, Double Byte and Localization Solutions, Software Quality Assurance & Contract Programming Services. We've also hit bulls eye in AQUA implementation, CARBON porting, QuickTime development and Japanese Language implementations. So, if your aim is to get the best software development for **Mac OS X**, target the experts.



RECOSOFT CORPORATION

RecoSoft Corporation, Osaka, Japan. Tel.: +81-6-6443-0015 Fax: +81-6-6443-1458 E-mail: info@recoSoft.com Website: www.recoSoft.com



makers of

PERIPHERALS, EXPANSION CARDS & SMART DEVICES

get OS X compatible

Since 1991, high tech companies like Apple Computer, Hewlett Packard, and Leica Microscopy have turned to Art & Logic for assistance in implementing cutting edge technologies.

Now, with the advent of OS X, Art & Logic is helping companies make their products compatible with Apple's new operating system. Art & Logic engineers are industry leaders in Carbon & Cocoa porting, Aqua implementation, and driver development.

In the new economy, where time to market is everything, you need results. Art & Logic delivers.

"We have found the engineers at Art & Logic to be outstanding—better than excellent. When we use their software development services, we get results."

Ted Dykes, CEO of LRO, Inc.



Art & Logic, Inc.

www.artlogic.com
877-278-5644 (toll free)
info@artlogic.com

The software engineering company that helps bring your hardware product to market—guaranteed.

Programming at the Speed of Light

Aqua

Cross-Platform

Device Drivers

TCP / IP

Graphics

Plug-Ins

Mac/Unix/Win32

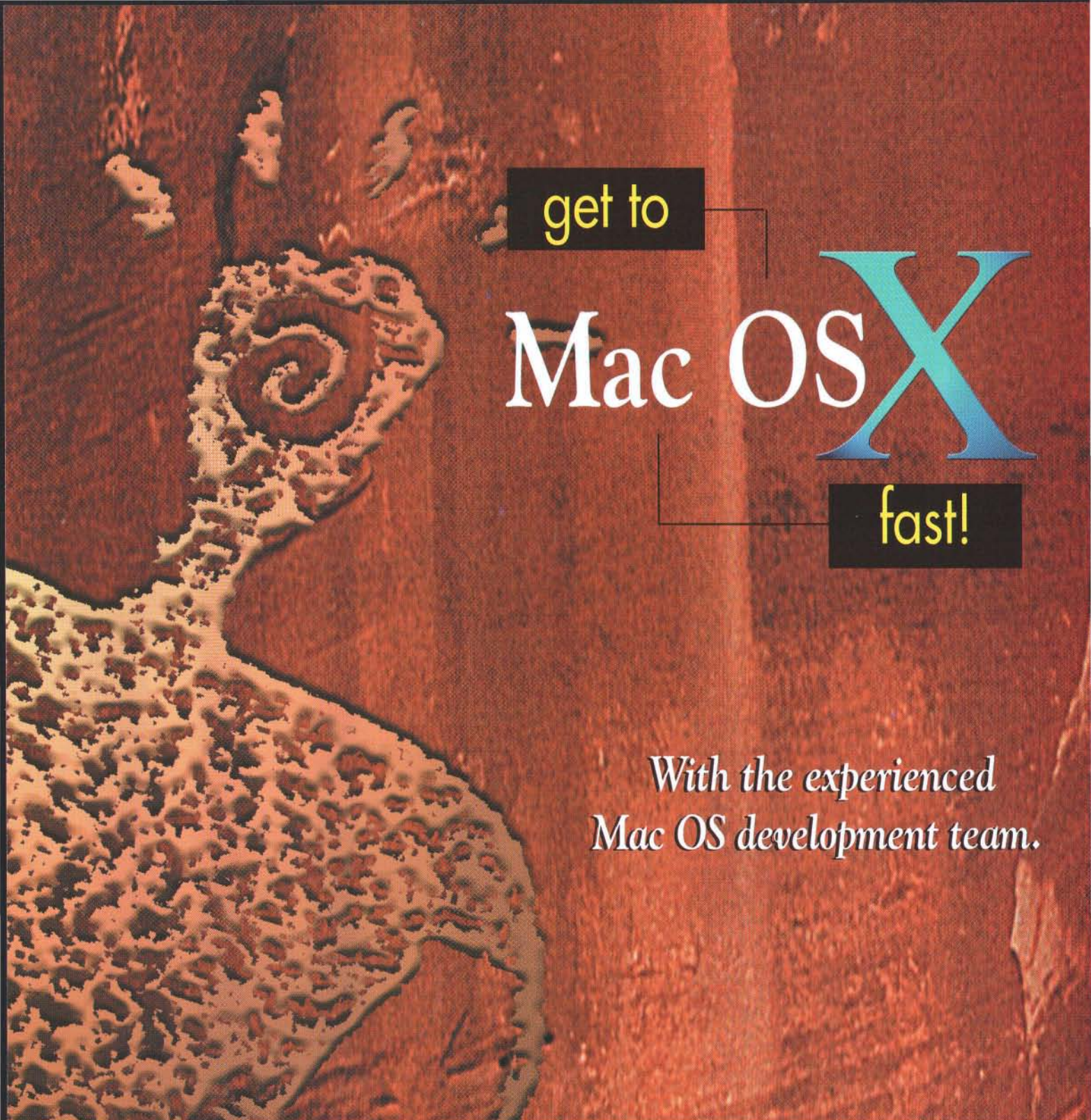
Ports

Carbon / OS X



Software Development
Outsourcing Since 1990

415.491.2200
www.shadetreeinc.com



get to

Mac OS X

fast!

*With the experienced
Mac OS development team.*

Red Rock Software specializes in the Macintosh software development. Red Rock Software's team of senior engineers averages over 10 years of development experience on the Macintosh platform. Our expertise and experience has gained us the trust of companies like *Apple Computer, Iomega Corporation and Sorenson Media.*

R E D R O C K

s o f t w a r e

If you are looking to get your product compatible with Mac OS X quickly and with extreme quality, let our experts help.

- Aqua Interface Implementation
- OS X Carbon Porting
- OS X Native Cocoa Application Development

call Red Rock at **888.689.3038**
or visit us online at www.redrocksw.com

- ☐ Eat your vegetables.
- ☐ Exercise every day.
- ☒ Port to Mac OS X.
- ☐ Call your mom.

All of these are good for you.
We can make one of them easy.

Since 1989, The Omni Group has worked with the technologies that have been refined into Mac OS X.

Moving to Mac OS X is a big step for your company, and you need consultants who can help you both plan how best to make the transition and follow whatever path you choose.

That's been our business for years. No matter what kind of product you have, we can get it up under OS X, fast:

- Real games: We ported id's Doom and Quake games to NEXTSTEP and OS X. Quake 2 took us a week.
- Big apps: We ported Adobe's FrameMaker to NEXTSTEP and Sun's Concurrency to OpenStep/Solaris.
- Big libraries: We ported the Oracle 8 client libraries which Apple ships today in OS X Server.
- Serious drivers: We ported 3dfx's Glide and wrote Voodoo2 and Rendition drivers for OS X Server.
We've written new mouse drivers for OS X Server and joystick drivers for OpenStep.
- New apps: We wrote OmniWeb, the only native OS X web browser, and OmniPDF, the native Acrobat viewer for OS X.

Mac OS X is what we do. Let us help you do it, too.

THE OMNI GROUP



2707 Northeast Blakeley Street
Seattle, Washington 98105-3118
www.omnigroup.com/consulting

sales@omnigroup.com
800.315.OMNI x201
206.523.4152 x201

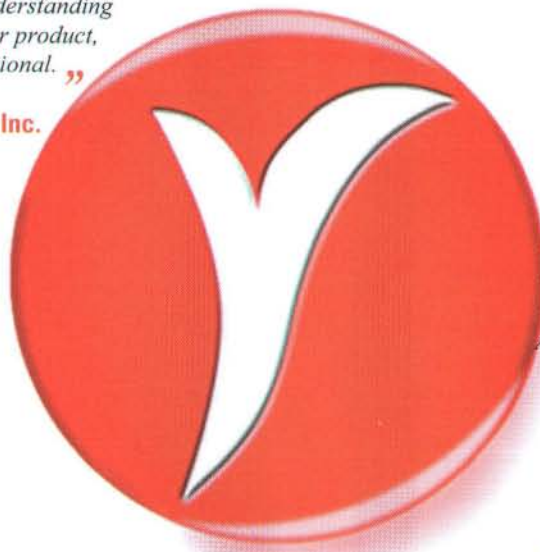
Robosoft has grown to be one of the most experienced Mac product development outsourcing companies. Here are a few reasons why ...

“ Robosoft's best of class development practices help them deliver the impossible --
high quality solutions, on budget, on time.
They are one of the finest Mac developers we have in Asia and
Apple is proud to be their partners in progress. ”

Eshwar Vangala, Apple Computers.

“ Robosoft completely rewrote my software application
exceeding my every expectation. In terms of
attention to technical detail and understanding
the business issues of the NetJumper product,
the Robosoft team were exceptional. ”

Gilbert Borman, NetJumper Inc.



After working with Robosoft on Mac VersaBook
project, we have no doubts that we will want
to continue working with them in the future.
Their **attitude was always positive** and did
anything that we needed to help us get
the product released. ”

Jake Benjamin, Versaware Technologies.

robosoftin.com

“ Robosoft Technologies receives my very highest recommendation. Prior to
awarding Robosoft my current ASP project, I had never met the staff and had
great concerns about distance and communications. My worries turned out to
be unfounded. Robosoft could not have been **more responsive** if they were in
the next office. Although I have yet to meet them personally, I think of
Robosoft, not as a contractor, but as my IT partner. ”

Barry Shapiro, The Anderson Group.

**Partners
in Product
Development**

**info@robosoftin.com
+91>8252>35458**

There are many cable testers on the market, but the CT-2 has one essential advantage: it splits into two pieces, allowing you to test cables whose ends are no where near one another. Additionally, the CT-2 can test connections through other devices. For example, I recently used the CT-2 to debug a problem I was having installing a new Ethernet jack in the wall. Through repeated uses of the CT-2, I was able to quickly isolate the problem, which turned out to be a faulty Ethernet port in one of our hubs! At one point I had one end of the CT-2 connected to one end of a patch cable, which connected to the patch panel, which connected to Category 5 wire running through the walls to the jack that I had just installed, to which another patch cable was connected, on whose other end the remainder of the CT-2 rested. Clearly, without the CT-2 I would have been helpless.

MacSense has just released a new and improved version of the CT-2, known as the CT-210, which swaps the older serial ports for USB ports. The essential functioning of the device remains the same. If you ever have to do network troubleshooting, save yourself a lot of frustration and pick up a CT-210 cable tester from www.devdepot.com.

CLEANER 5

If you spend any time at all working with multimedia files, Terran Interactive's latest release of Cleaner will quickly become one of your favorite applications. Especially useful for creating streaming multimedia files, Cleaner is a powerful program that can be equally appreciated by the high-end system administrator for making streaming QuickTime or RealSystem movies, or by the enterprising college student ripping MP3s in the wake of the Napster fallout.

As you can see from the list of supported file types below, Cleaner is about as versatile as they come. In addition to merely converting these files, Cleaner gives you the ability to tack on a few bells and whistles as well, such as adding fade in/fade out to audio or movie files, or blurring/sharpening image files. Aside from out and out editing of the actual content of a file, Cleaner is a fully functional application for putting the finishing touches on any media project.

DV	Reads	Writes
QuickTime	Reads	Writes
WAV	Reads	Writes
PICT, JPEG, PNG, BMP	Reads	Writes
TIFF, GIF	Reads	-
RealSystem	-	Writes
Windows Media	-	Writes
Video for Windows (AVI)	Reads	Writes
MPEG-1	Reads	Writes
MPEG-2	Reads	Writes
MPEG Layer I and II audio	Reads	Writes
OpenDML	Reads	-
Macromedia Flash	Reads	-
MP3	Reads	Writes
AIFF	Reads (including importing audio from CDs on Mac)	Writes



Don't let this happen to you.

Stay suited up with The Regulator Pro™ UPS for Macintosh® Computers from Belkin.

The Coast Guard calls it *Semper Paratus*—always prepared. To us, it's the **Regulator Pro Gold Series**. Belkin knows you can't afford to risk your data and hardware to unpredictable forces like blackouts, brownouts, spikes, and surges. With a Data Recovery Warranty, a Connected Equipment Warranty, and up to 45 minutes of battery back-up time, as well as the great features listed below, we're here to make sure you won't get caught with your... uh, guard down.

- Powerful 500VA and 650VA models available
- Automatic Voltage Regulation (AVR)
- Ships with free shutdown management software
- Compatible with PC or Mac® OS 9.0.4 and higher
- 8 AC outlets



Belkin Components • 310.898.1100 • Fax 310.898.1111
Compton, CA • Atlanta, GA • United Kingdom • Holland

All rights reserved. All trade names are registered trademarks of respective manufacturers listed.
©2000 Belkin Components. 20AD358

Coming to a MacTech near you...

ADC Direct

*A regular update from the
Apple Developer Connection*

Apple Developer Connection will bring you regular updates on technical issues, resources available, and insights from those in the know at Apple about the best ways for developers to work with Apple, and take advantage of what they have to offer. This new section will be featured in every issue of MacTech, starting with January, 2001!

Catch the maiden voyage of this new section in the January, 2001 issue!



| A | D | C |

Apple Developer Connection

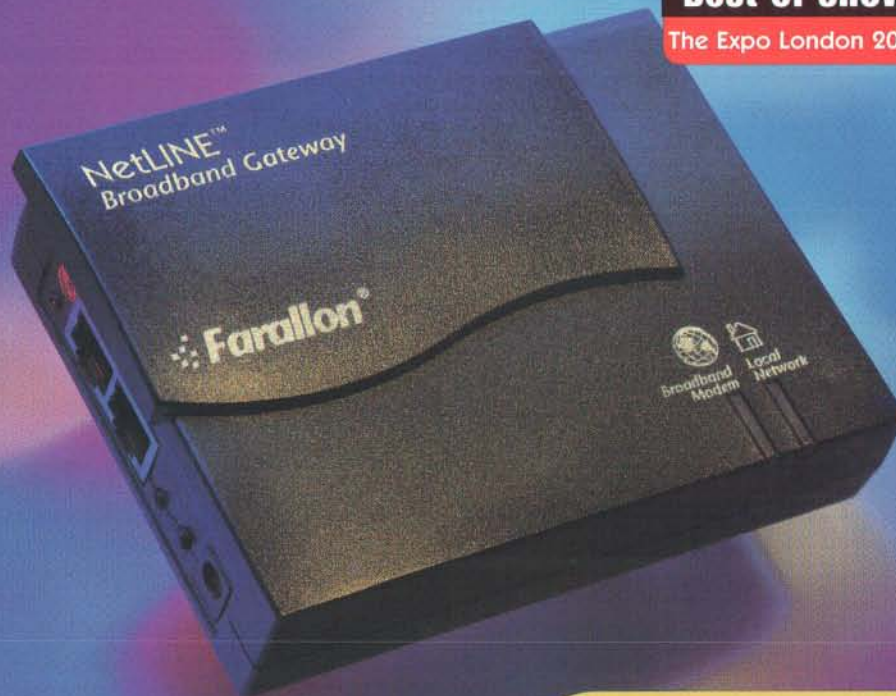
MacTech

Share your high-speed Internet connection and protect your data from unauthorized access with the

NetLINE™ Broadband Gateway **Farallon** www.farallon.com

Best of Show

The Expo London 2000



*The perfect solution for
home, office or school!*

The NetLINE Broadband Gateway

allows you to share a high-speed Internet connection, such as Cable or DSL, with multiple computers using just a single IP address from your ISP.

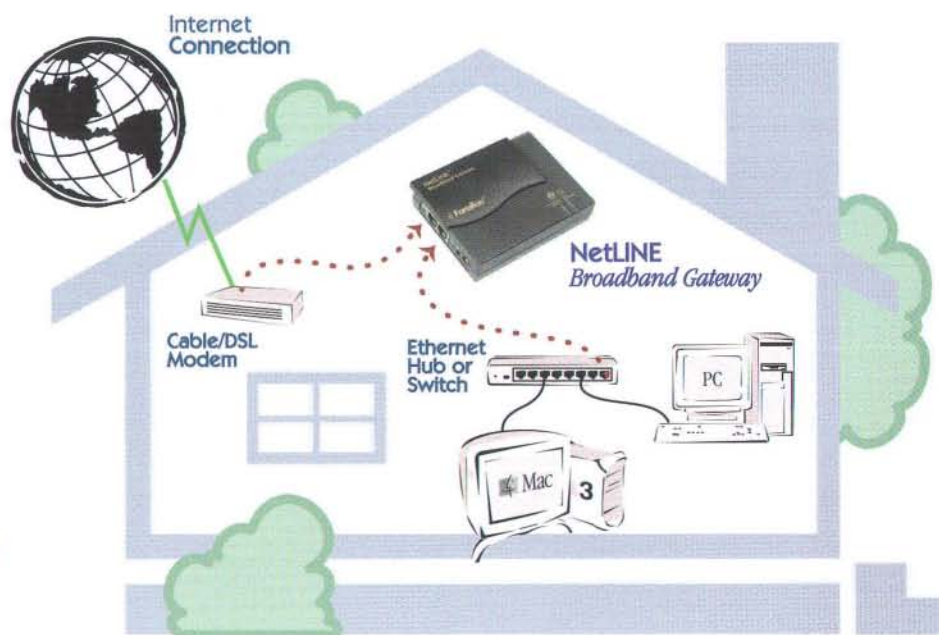
The built-in firewall will protect your important files by preventing unauthorized access via the Internet.

Works with both Macs and PCs. Web-based configuration makes set-up a breeze.

**Receive a coupon for
25% off a Farallon hub
or switch when you
purchase the NetLINE
Broadband Gateway.** (US Only)



For more information contact
Dr. Farallon at 1-800-613-4954 or
visit us on the web at www.farallon.com



The NetLINE Broadband Gateway connects between
your network and your Cable/DSL modem

Available
at **the DEPOT**



BOOT. SERVE. RUN. RUN. RUN ...
INTRODUCING CHAMPION SERVER 1.2.1

COPYRIGHT 2000, TERRA SOFT SOLUTIONS, INC.

WWW.YELLODOGLINUX.COM
BY TERRA SOFT SOLUTIONS, INC.

One limitation that I ran into when using Cleaner occurred when I tried to burn an Audio CD of AIFF files exported from Cleaner using Adaptec Toast. The AIFF files exported by Cleaner were compressed, and Toast didn't want to burn them. I quickly found a simple work-around, however, by using SoundApp (a tiny shareware application) to convert Cleaner's AIFF files to SoundApp's AIFF files, which Toast happily accepted. (Note that SoundApp alone is not a viable solution, since it only reads WAVs, AIFFs, and Mac System Sound files.) Unfortunately, Cleaner also won't read RealAudio documents, so converting all those Grateful Dead shows you downloaded from <ftp://ftp.deadshow.com/> to MP3 won't work either. I know, I know...it hurts me too.



Figure 1. Cleaner 5 eats up MP3s but can't read RealAudio.

Still, Cleaner 5's batch processing capacity makes it a must if you are working with large numbers of files, and its easy to use interface makes choosing from its myriad options surprisingly easy. One additional feature that I did not get to try out is MotoDV, which allows you to capture DV directly from your FireWire compliant video camera. Terran promotes Cleaner 5 as a "complete camera-to-web" solution, which I don't see how they can justify when you consider Cleaner's inability to actually edit the content of movies (a la Final Cut Pro or iMovie). However, its Leatherman-like abilities mean that if you haven't already, you will greatly benefit from adding Cleaner 5 to your multimedia application arsenal. **M7**

• Custom
Destinations

• Electronic
Transaction
Processing

• Easy
Trialware
Creation

• Install or
UnInstall

Get It Together With InstallerMaker™

• Installation
From FTP or
HTTP Sites

When Time Is Money, Get It Done Fast!

Build Smaller Installers – With StuffIt InstallerMaker, you'll build installers faster than ever before! Compress your installers an average of 15% smaller using the power of the StuffIt Engine®. Smaller installers download faster, provide added space on CDs and servers, and increase network bandwidth.

Quickly Create Demoware – Easily turn your application into a polished demo. Just set the number of days for the demo to be active, paste in the graphics, and you're done!

Archive Freshening – Automatically update your installer project file; eliminate repeated searches for modified files.

• Resource
Installation

• Built-In
Resource
Compression

• ShrinkWrap
Disk Image
Support

Scripting Saves Time – InstallerMaker provides full scriptability for all features. Automating the build process saves time and helps ensure the integrity of your installer.

Trialware in Minutes – Creating trialware is a breeze with InstallerMaker. With just a few clicks, and no extra coding, you can create trialware that is e-commerce ready.

• Marketing
Opportunities
To Help You
Make Money

Update in a Flash – Easily build intelligent "diff" files in installers to create small updaters for quick online distribution. From one simple installer, you can update 68k, PowerPC or FAT applications.

• Hierarchical
Package
Support

More Details Online – There's a lot more InstallerMaker can do for you. Get all the info at www.aladdinsys.com.

• Built-In
Updaters



By Andrew C. Stone

What A Drag!

Creating drag wells in Cocoa

The richness of the user interface in Macintosh OS X is due, in part, to the abundance of elements that allow dragging and dropping of data within and between applications. For example, Cocoa uses drag and drop for applying color swatches to text selections and objects. Stone Design's flagship application, Create™, makes heavy use of the drag and drop metaphor. A library of resources allows users to drag in components, pages, effects, blends, images, and patterns. The user can quickly save a selection of graphics to one of many image formats with Create's "Image Well". This article will cover the basics of creating a custom control which allows a user to drag in a file, and then allows the user to drag out the file.



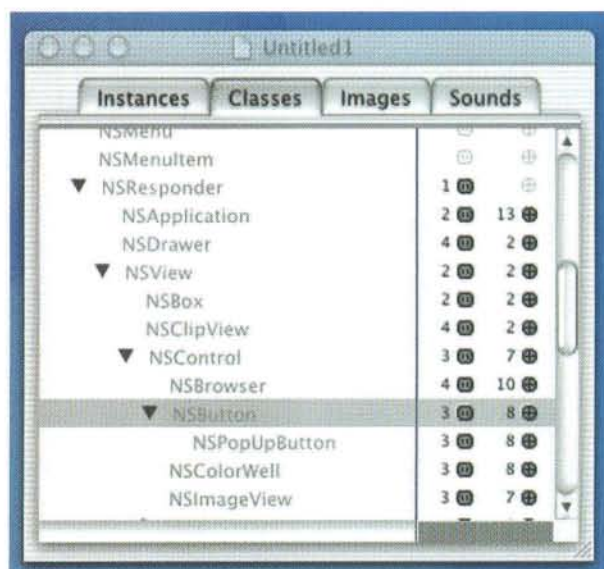
Create's image well lets you drag out data in various image formats.

First, a bit of article administrivia. Mac OS X has been a fast moving target with the various developer releases, Public Beta, and soon OS X GM (for Golden Master which always reminds me of that Eddy Murphy movie about G)! Because of the changes introduced with each release and the several month lag time between penning articles and when MacTech hits the stands, there are times when what I've written doesn't jive with the current reality. To address this, I've added a section to the Stone Design website for MacTech article updates and errata. So if you get confused, check out www.stone.com/dev/MacTech_Errata/ to see if there are updates, and if not, email me so I can post fixes.

It's actually quite simple to create new types of user interface controls using Cocoa because of the well designed hierarchy of classes that comprise the Application framework. Your first job is to find an existing class that your control can inherit from, which will reduce the amount of code that you have to write. I keep `/Developer/Documentation/Cocoa/cocoa.html` open so I can quickly navigate to the AppKit or Foundation API's. But an even easier way to look at the object hierarchy is to use InterfaceBuilder. First launch ProjectBuilder and create a new Cocoa Application project named ImageWellTester. Click on the "Resources" triangle in the Files and Groups outline view, and double-click the MainMenu.nib. This will launch InterfaceBuilder.

Click on the "Classes" tab of the document window, and then explore the NSObject outline view:

Andrew Stone <andrew@stone.com> washes windows at Stone Design Corp <<http://www.stone.com/>> and divides his time between writing applications for Mac OS X and raising farm animals, including children.



Interface Builder lets you explore the hierarchy of objects offered by Cocoa.

Our drag well is definitely a subclass of NSView, but we can gain further functionality by subclassing NSControl, which gives us the target/action support. Select NSControl in the Classes pane. If you hold down the CONTROL key while clicking on NSControl, a context-sensitive menu appears. Do this and select "Subclass". A new class which inherits from NSControl is created and named, by default, MyControl. Rename the class to DragWell.

Why write code if you can have it written for you? You can use InterfaceBuilder to produce stub files where all you have to do is fill in the functionality in the stubbed out methods. Click on the "outlet" icon to add instance variables to the class, and the "+" icon to add actions to the class. For now, we don't have any to add. Be sure DragWell is selected, and choose select "Create Files..." from the Classes menu (or again using the context menu via CONTROL key). They will be named DragWell.m and DragWell.h - leave the "Insert into Project Builder" selected and click OK.

Drag out a "Custom View" from the IB palette window, second tab, onto to the main window. Choose Tools->Inspector->Custom Class popup button, and click on "DragWell". The custom view should now say "DragWell". Save the file.

Go back to ProjectBuilder, select the newly added DragWell.h and DragWell.m and you'll see the stub:

```
#import <Cocoa/Cocoa.h>

@interface DragWell : NSControl
{
}

@end
...
```

Scripter 2 with ScriptBASE

NEW!
Version 2.2

**Tap the power of AppleScript
with Main Event's Scripter!**

**For professionals and novices
Webmasters and solution providers**

**No matter what your experience,
Scripter makes it easier!**

BEGINNER AT SCRIPTING?

- Unique command-builders help you learn to assemble commands
- Built-in tools for experimenting
- Shortcuts to speed scripting

EXPERIENCED WITH APPLESCRIPT?

- Unmatched single-step interactive debugging
- Watch and modify global and local variables while stepping
- Debug messages sent to script applications
- Includes ScriptBase to integrate scripting scenarios

*Scripter received MacWeek's 5-diamond rating – Twice!
"Scripter's virtuoso display of AppleScripting savvy and
debugging wizardry make it the best environment we've
found for learning or creating AppleScript scripts."*

—MacWeek

**Make AppleScript and your
applications work for you!**



Main Event
PO Box 21470
Washington, DC 20009
Tel: 202-298-9595
Sales: 800-616-8320
info@mainevent.com
www.mainevent.com

Macworld

Conference & Expo™

ahead

To Request
More Information,
Visit www.macworldexpo.com!

Moscone Convention Center
San Francisco, CA
January 9 – 12, 2001

Register Online

www.macworldexpo.com

Call Toll Free 1-800-645-EXPO

Refer to Priority Code: A-MTCH

World-Class Exposition!

**Macworld Conference & Expo/
San Francisco 2001 is a
One-Stop-Shop offering discounts
not found anywhere else!**

- Featuring more than 500 exhibiting companies
- Thousands of new products, software and services
- Hands-on demonstrations
- Test-drive new technologies
- Awards and competitions
- Prizes, drawings, giveaways

Special Interest Boulevard

**Macworld Conference & Expo offers
so much for every Mac user!**

Check out the Special Interest Boulevard while you are exploring the exhibit floor to find and compare hot new products for your particular needs. Popular areas include:

- Digital Media
- Small Business
- Sci-Tech
- Education/Assistive Technology
- 3D
- Edutainment

**Be sure to visit www.macworldexpo.com
for exciting additions!**

**Join your friends
and colleagues in the Mac community
at The Largest IT event
on the West Coast!**

Cutting-Edge Educational Programs!

**Conference sessions for the
New, Beginning, Intermediate and
Advanced users!**

Macworld/Pro features technically in-depth presentations and issues-oriented discussions about professional applications of Macintosh technology. The Pro conference offers **the most sophisticated training available** on Mac networking, digital video, art director/creative management practices, and Mac systems administrations for large organizations.

Macworld/Users continues to be one of **the best educational values anywhere.**

The Users conference features industry experts offering skill development on the most popular Mac-related tools and professional development courses for users who rely on Apple technology to gain a competitive advantage.

MacBeginnings – San Francisco debut!

After great success this July in New York, MacBeginnings will make it's debut in San Francisco! These high-energy, informative sessions will provide first-time attendees, new and beginning Mac users a starting point to enter the Mac community. MacBeginnings are free and open to **ALL** registered Macworld Conference & Expo attendees!

Macworld
Conference & Expo.

www.macworldexpo.com

Flagship Sponsors

Macworld

Macworld.com

MacWEEK.com

MacCentral.com

Owned and Managed by

WORLD EXPO

```
#import "DragWell.h"

@implementation DragWell

@end
```

Every custom view must define its own drawing method, `drawRect:`, where the actual drawing of the view takes place. We'll want a simple, depressed beveled look, with an icon to drag in the center. And our control will adhere to two protocols - `NSDraggingDestination` and `NSDraggingSource`, to allow both dragging out of info and accepting dragged in info. A complete list of the methods required by the protocols are found in:

/System/Library/Frameworks/AppKit.framework/Headers/NSDragging.h

Here's the 'well' commented code for the `DragWell` - type it in (or download it from www.stone.com/dev/MacTech/Jan-2001/). You can copy and paste it as a template for use in other objects, modifying where needed to copy other data types to the drag pasteboard as needed.

```
+++++++ DragWell.h ++++++

#import <Cocoa/Cocoa.h>
```

```
@interface DragWell : NSControl
{
    NSImage *image; // the image displayed and dragged by
the user
    NSString *file; // the file path that the user will drag out
    unsigned int last; // for optimizing the draggingUpdated:
method
}

// if you want to programmatically set a file in the well:
- (void)setFile:(NSString *)newFile;

@end

+++++++ DragWell.m ++++++

#import "DragWell.h"

/* Example source by Andrew Stone andrew@stone.com
www.stone.com for MacTech */

@implementation DragWell

// when our DragWell is reconstituted from the nib file, awakeFromNib
gets called

- (void) awakeFromNib
{
    // this is where you register for various types of drag pasteboards:
    [self registerForDraggedTypes:[NSArray
 arrayWithObject:NSFileNamesPboardType]];
}

// Never add an ivar which allocates memory without releasing it in
dealloc:
// Don't Litter!

- (void)dealloc {
    [file release];
    [image release];
    [super dealloc];
}

// Every custom view must implement drawRect:
// We'll draw our depressed beveled rect
// and if there is a file, we'll draw an image for dragging out

- (void)drawRect:(NSRect)rects
{
    // one simple call draws the depressed bezel:
    NSDrawGrayBezel([self bounds] , [self bounds]);

    // if you like the pinstripes, then include these two lines:
    [[NSColor windowBackgroundColor] set];
    NSRectFill(NSInsetRect([self bounds], 3.0, 3.0));

    // find the center of the view and draw the image over:
    if (image != nil) {
        NSPoint p;
        NSSize sz = [image size];
        p.x = ([self bounds].size.width -
        sz.width)/2.;
        p.y = ([self bounds].size.height -
        sz.height)/2.;
        [image compositeToPoint:p
        operation:NSCompositeSourceOver];
    }

    // When we set the file, we also set the image which we get from the
    NSWorkspace:

    - (void)setFile:(NSString *)newFile {

        // it's good practice to not do any work if nothing would change:
        if (![newFile isEqualToString:file]) {

            // live clean and let your works be seen:
            [file release];
            [image release];

            // copy the newFile into our instance variable file:
            file = [newFile copyWithZone:[self zone]];
        }
    }
}
```

DATA RECOVERY: 800-440-1904

"Their incredibly kind staff focus on getting the job done as well as their customer's feelings. All experiences should be so pleasant."

—Neil Tickin, Publisher
MacTech Magazine

7 Good Reasons to Choose DriveSavers



1. Fastest, most successful data recovery service available.
2. Retrieve recovered data instantly with DATAEXPRESS™ over high speed secured Internet lines.
3. Authorized to perform Data Recovery on all Apple computers and hard drives.
4. 24-hour, onsite, and weekend services.
5. Advanced, proprietary recovery techniques.
6. Featured by CNN, BBC, Forbes. Also in Mac World, Mac Addict, Popular Mechanics, and many others.
7. Federal and State Contracts (GSA, CMAS).



"We Can Save It!"

INTL: 415-382-2000
www.drivesavers.com

DriveSavers – Your Data Recovery Solution!

©2000 DRIVESAVERS, INC. 400 BEL MARIN KEYS BLVD., NOVATO, CA 94949, INTL: 415-382-2000, FAX: 415-883-0780

```

// check if the file is nil or empty string:
if (!file || [file isEqualToString:@""]) {
    image = nil;
} else {
    // we have a file, now let's find the image for that file:
    image = [[[NSWorkspace sharedWorkspace]
iconForFile:file]retain];
}

// we'll need to redraw next event
// this is the approved Cocoa way of setting ourselves "dirty"
[self setNeedsDisplayInRect:[self bounds]];
}

// here we copy the filename to the pasteboard
// I like to factor this out because it may vary from object to object:
- (BOOL)copyDataTo:pboard
{
    if (file != nil && ![file isEqualToString:@""])
    {
        [pboard declareTypes:[NSArray
 arrayWithObject:NSFileNamesPboardType] owner:self];
        [pboard setPropertyList:[NSArray
 arrayWithObject:file] forType:NSFileNamesPboardType];
        return YES;
    } else return NO;
}

// DRAGGING SOURCE PROTOCOL METHODS
// To add drag away functionality to a control, implement these:

// draggingSourceOperationMaskForLocal lets you control the behavior
of what should
// happen with the data on the pasteboard: copy, link or any
// the isLocal flag tells you whether the object querying is from within
your app
// or from another application running on the system

```

```

- (unsigned
int)draggingSourceOperationMaskForLocal:(BOOL)isLocal
{
    if (isLocal) return NSDragOperationCopy;
    return
NSDragOperationCopy|NSDragOperationGeneric|NSDragOperati
onLink;
}

//The simple dragImage:at:offset:event:pasteboard:source:slideback:
method
// is all we do to initiate and run the actual drag sequence
// But we only do this if we have an image and we successfully write
our data
// to the pasteboard in copyDataTo: method

- (void)mouseDown:(NSEvent *)e
{
    NSPoint location;
    NSSize size;
    NSPasteboard *pb = [NSPasteboard
pasteboardWithName:(NSString *) NSDragPboard];

    if (image && [self copyDataTo:pb]) {
        size = [image size];
        location.x = ([self bounds].size.width -
size.width)/2;
        location.y = ([self bounds].size.height -
size.height)/2;

        [self dragImage:image at:location
offset:NSZeroSize event:(NSEvent *)e pasteboard:pb
source:self slideBack:YES];
    }

// DRAGGING DESTINATION PROTOCOL METHODS
// To add drag acceptance functionality to a control, implement these
methods:

```

Your ONE-STOP Price Comparison for Movies!

MOVIE DEPOTsm

www.moviedepot.com

```

// this is called when the drag enters our view
// by returning NSDragOperationCopy
- (unsigned int) draggingEntered:sender
{
    NSPasteboard *pboard;

    last = NSDragOperationNone;
    pboard = [sender draggingPasteboard];

// we don't accept drags if we are the provider!!
    if ([sender draggingSource] == self) return
    NSDragOperationNone;

    if ([[pboard types]
containsObject:NSFileNamesPboardType]) {
        if (image == nil) {
            image = [[[NSWorkspace sharedWorkspace]
iconForFile:[pboard
propertyListForType:NSFileNamesPboardType]objectAtIndex:
0]]retain];
            [self setNeedsDisplayInRect:[self
bounds]];
        }
        // we'll copy or link depending on the intent of the dragging source:
        last = [sender draggingSourceOperationMask];
    }
    return last;
}

// instead of constantly rechecking the pasteboard as the mouse moves
inside the view
// we'll simply return the cached value that we set in 'last' in
draggingEntered:

- (unsigned int) draggingUpdated:sender
{
    return last;
}

```

USBStuff

1-88-USB USB US
-or- (1-888-728-7287)

WorldWide Distributors of USB
and FireWire Parts,
Peripherals and Accessories

Hey Developers:

<http://www.usbstuff.com/developers.html>

1-877-4 HOTWIRE
-or- 1-877-446-8947

FireWireStuff

```

// Because we're providing feedback by setting the file right when the
user enters
// we'll need to undo that work if the user does not let go of the drag
inside and exits instead:

```

```

- (void) draggingExited:sender
{
    // the user has exited -> clean up:
    if ([sender draggingSource] != self) {
        if (file == nil) {
            // then unset the file image we set in mouseEntered as
            feedback...
            [image release];
            image = nil;
            [self setNeedsDisplayInRect:[self
bounds]];
        }
        last = NSDragOperationNone;
    }
}

```

```

// any dragging clean up might be done here
// don't forget to return YES!

```

```

- (BOOL) prepareForDragOperation:sender
{
    return YES;
}

```

```

// Actually do the work in this method if it's not too time consuming
// Otherwise, you may consider returning YES, and doing the work
// in concludeDragOperation to prevent the drag from sliding back
// because it "timed out"

```

```

- (BOOL) performDragOperation:(id
<NSDraggingInfo>)sender
{
    NSPasteboard *pboard;
    pboard = [sender draggingPasteboard];
    [self setFile:[pboard
propertyListForType:NSFileNamesPboardType]objectAtInd
ex:0]];
    return YES;
}

```

```

- (void)concludeDragOperation:(id
<NSDraggingInfo>)sender
{
    // we already did the work in draggingEntered
    // You might notify some other object that the file is here
}

```

```

// this is good if you want to be able to drag out data even if the
window is not
// front most, the first click will do more than just bring the window to
front -
// It will also allow the drag to begin on that first mouse down

```

```

- (BOOL)acceptsFirstMouse:(NSEvent *)theEvent {
    return YES;
}

```

@end

CONCLUSION

Adding drag and drop support to controls and views is very easy in Cocoa. Because ease of use and intuitive behavior is the keystone of Mac OS X, be sure to make your applications conform to this model.



You'll be stuck in San Francisco's heaviest traffic and loving every minute of it.

At Macworld Expo SF 2001, we make it easier for all techies, novice and expert, to have a space as special as their ideas. Formerly Developer Central and Net Innovators, MacTech[®] Magazine presents MacTech^(sm) Central — an exhibit at Macworld that showcases technical and developer tool/service companies for the Macintosh in the biggest possible way. Macworld takes place in San Francisco, January 9-12, 2001. Check out the exhibits at the show (Booth 3240) and on the web at: www.mactechcentral.com.



MacTech is a registered trademark of Xplain Corporation. MacTech Central is a service mark of Xplain Corporation.
Other trademarks and copyrights appearing in this printing remain the property of their respective holders.

by Jeff Clites <online@mactech.com>

The New Basics

What Mac user out there doesn't remember Apple's famous 1984 commercial? It only aired once, during the 1984 Super Bowl, but it won numerous awards and is held by some to be the greatest commercial of all times. It was directed by Ridley Scott, director of *Blade Runner* and more recently *Gladiator*. I still get chills every time I see it. The interesting thing to remember is that when that commercial came out, Big Brother was IBM — they were the competition, the bad guys, the *enemy*. Now, the villain might be Microsoft (if anyone), and IBM is, well, just there. They seem like neither enemy nor ally, just more of a quiet neighbor — you know they're around, but you're not sure what they do. They've reinvented themselves, and there's almost no overlap between their target market and Apple's. But the landscape of computing is changing, and there's a party going on. It's more of a Rave than a party, actually — there's all kinds of strange stuff brewing, and lot of people are going but nobody's really talking about it during the day. The theme of the party is Unix, in a way, but it's more than that. Suddenly, Unix underlies every important operating system out there — except Windows. It's the basis of Mac OS X, Linux, BSD, and all of the heavyweight server software by IBM, Sun, SGI, and HP, and even smaller niche OSes like QNX Neutrino. This doesn't mean that we are all going to band together to fight Microsoft, because the new world isn't about fighting with Microsoft. It isn't about something so negative. The real point is that there's an incredible landscape wide open for the cross pollination of technologies, and this exchange isn't dependent on explicit alliances, but rather on standards — on just having things in common. It's an unconscious union. This is important — it means that a flow of ideas will happen even in the face of competition, because company A is better off if its software and hardware can talk to company B's than if it can't, and vice versa. And it's just icing on the cake when they are enlightened enough to share implementations rather than just specifications — to realize that it's to everyone's advantage to just share the code rather than waiting around for everyone else to reimplement it. This is what the Open Source movement is all about, from a purely practical perspective at least. This isn't a completely new thing of course, but it's new that the Macintosh will be fully a part of this standards-based, Unix-centered community, and it's new that this community will include just about everyone — and this is going to continue to push us forward as a whole.

1984 Commercial

<http://www.tbwachat.com/product/historical_work/tv/1984/1984.html>

Apple's 1984: The Introduction of the Macintosh in the Cultural History of Personal Computers

<<http://www.duke.edu/~tlove/mac.htm>>

So what does this all mean to the Macintosh developer? At the most obvious and literal level, it means that there's a bunch of software out there which will suddenly run on Mac OS X with little or no modification, ranging from applications to frameworks to libraries to individual routines — things directly useful to the consumer, to the system administrator, or to the developer. But more fundamentally, it means that there's a new set of basics out there — a new range of technologies that every developer needs to know about. And really, it won't be enough to just know that these technologies exist — developers will have to know how to use them as a matter of course. And these New Basics are new in two ways: there are technologies which will be new to Macintosh developers (and they may be new to everyone), and it's new that the essential basics are cross-platform — they're the same for almost all developers, independent of operating system or type of development. So this month, we're going to touch on the first of these, XML, and point you to my picks for the essential references. In the coming months, I hope to cover the rest of the New Basics. And when we're done, pass the list on to your Unix and Windows friends — they'll need them too.

By the way, you may be wondering why I singled out IBM to start off the discussion. They came to mind because their developerWorks site gives great coverage of the New Basics, and I'm constantly lead back to it. IBM has gone from a threat to a resource. Visit their site frequently to find ongoing information about important technologies.

IBM developerWorks

<<http://www.ibm.com/developer/>>

XML

The first New Basic is XML. As a technology, it has been hyped, over-hyped, and re-over-hyped in the press. It doesn't solve every problem under the sun, as some have implied, but it has done one important thing: it has gotten everyone thinking about information formats, and given us a common starting point to flock around when

designing new ways to store and interchange data, which makes it more likely that the formats we settle upon will be sensible and reasonably easy to process by anyone who wants to. For better or worse, it is being used just about everywhere possible, and this trend is only going to grow for the foreseeable future. Developers will have to be comfortable using it when reading and writing documents and when writing applications which talk to other applications, either locally or over the internet (which includes almost all applications being written today). So here are my picks for the essentials: First, the annotated XML specification on XML.com. It's the specification itself, with additional notes added by Tim Bray, one of its authors. It isn't what you'd want to use to learn about XML for the very first time, but it's very handy to have around when you need the final word on whether some piece of syntax is legitimate, and it can be quicker to just go to the source rather than searching for someone else's explanation. But when you're learning about XML as a developer, you're ultimately more interested in *how* than *what* — you need to process XML, and the specification doesn't cover strategies for this. The two best resources I've found are books. *Professional XML* (ISBN: 1861003110) weighs in at over 1000 pages, and it gives coverage to almost every major technology surrounding XML. Some of these technologies have evolved since the book was published almost a year ago, but it's still relevant to answer the "what's that?" and subsequent "okay, now how do I use that?" questions. A newer reference is *Essential XML: Beyond Markup* (ISBN: 0201709147). It covers a more focussed range of topics, but as a result it gives thorough coverage of all of the areas on which new technologies are likely to be built — methods for processing, specifying, navigating, linking, and transforming XML documents. And this book is refreshing and unique in two ways: For one, it focuses on the XML Information Set (Infoset), which gives the abstract definition of XML, independent of its syntactic detail. The Infoset almost never gets coverage elsewhere, but it's really the most

important specification, and it makes explicit the notion that what we normally think of as XML is just one way to write it down on paper, just one syntax out of many possibilities for serializing XML data. In the push for standards, this distinction is being overlooked, and there's an undue focus on the syntax of XML over its concepts — people are forgetting that the angle brackets aren't what's important, and an alternative syntax might be a good thing to have around. The other refreshing thing about *Essential XML* is that it explicitly acknowledges that XML has been over-hyped, and in fact the authors wrote it to help them sort out what really is important to the programmer.

Finally, my last essential reference for XML is the xmlhack web site. It's a news portal site, which gives summaries and links to relevant new bits of XML information on the internet. Just glancing at it from time to time will give you a feel for what areas are important and evolving, in addition to giving you pointers to new developments in technologies of specific interest. (I should mention too that these pointers will sometimes lead you to the XML.com site mentioned above, which is actually a great place to look for in-depth articles.)

The Annotated XML Specification

<<http://www.xml.com/pub/axml/axmlintro.html>>

xmlhack: developer news from the XML community

<<http://www.xmlhack.com/>>

XML.com

<<http://www.xml.com/pub/>>

So, that's it for the first of the New Basics. Remember to check back in the coming months, when I hope to cover more of them. Until then, I'll keep you in suspense about the rest of the technologies on my essentials list....

MT



C++ Developer's Kit

PROFESSIONAL

Advanced Tools for ANSI C++ Development

Suite of tools and reference collection showing developers how to create better ISO/IEC C++ code using UML&C++. It includes libraries of useful programming snippets and classes that you can use in your own projects to create more efficient powerful and reliable Mac applications.



"C++ Developer's Kit offers a complete set of tools to build software conforms to ANSI/ISO C++ standards. The package also helps solve problems that might emerge in development. The software explains the fundamentals of UML and OO development with a variety of models and applications."

Institute of Electrical & Electronics Engineers

Try out free C++ sample code at: <http://www.technosoftweb.com/C++>

TECHNOSOFT

mailto: orders@technosoftweb.com

www.technosoftweb.com

From C to C++ more effectively!

C++ Developer's Kit Professional is a comprehensive C++ development kit for developers who need powerful C++ and UML features. This new development kit includes ISO/IEC C++ based examples for CodeWarrior, MPW, Symantec C++. This kit offers examples, projects, classes, functions libraries, templates, algorithms, tools and much more. It also provides over 250,000 code lines, the complete CAD++ library with classes and functions to design CAD applications, Garbage Collection-based classes, nested classes, UML/C++ models and more!

INSIDE THE KIT

3 CDs
2 Manuals
1 UML/C++ Quick Reference
C++ guide
C++ tools
C++ snippets
C++ projects
C++ classes
C++ Web sites
UML standard
UML seminar
UML models
STL library
OOP topics
Software Reuse

COMPUTER

developer's
C++ news
Technical News for Software Developers

CodeWarrior
LITE version INSIDE

DHL

FREE shipping
(For Europe-CEE, USA and CANADA)

Just \$189 from
@DEPOT

Continued from page 4

statements, and comments. Chapter 7 focuses on functions, but contains some information on macros as well. Chapter 8 shows how C++ programs can be modularized with the help of namespaces and introduces exception handling. Chapter 9, finally, is about organizing C++ programs into files.

All the chapters are rather terse with mostly tiny code examples. (A notable exception is the desk-calculator example in the first section of the sixth chapter.) Since the objective is to introduce language features, and not to teach how to program, I do not think that this is a problem. On the contrary, readers familiar with C-like syntax will appreciate Stroustrup's style of quickly getting to the point. Others should not have any problems, either, even though they might need to read a section more than once in order to familiarize with the C++ syntax.

ABSTRACTION MECHANISMS

The six chapters of part II describe the language constructs that support abstract data types (ADTs), object-oriented programming (OOP), and generic programming.

Chapter 10 explains how ADTs can be defined that in their "look-and-feel" do not differ from built-in types. Chapter 11 complements chapter 10 by introducing operator overloading as well as friend functions and conversion operators. Both chapters lay the foundation of the rest of the book, in particular of chapters 12 and 15 that treat C++'s support for OOP and also introduce the OOP vocabulary.

Chapter 13 and chapter 14 cover templates and exception handling, respectively.

These chapters are less terse, and the code examples are bigger as well. Readers with experience in OOP will again be happy with the pace. However, I am afraid that many OOP novices will need to look for a more gentle introduction to OOP.

THE STANDARD LIBRARY

The seven chapters of part III present the standard library. Unlike the other four parts, I have not re-read this part for this review. (I have read the book [albeit the 1st printing] in its entirety about three years ago and have used it as a reference manual since then.)

But I remember that I had no difficulty to understand the organization and semantics of the library. Furthermore, whenever I look for a certain stream or container, I quickly find it and am able to put it to good use in my code.

DESIGN USING C++

The three chapters of part IV, *Development and Design*, *Design and Programming*, and *Roles of Classes*, round off the book.

This part cannot be a replacement for a good software-engineering book on, for instance, modeling, patterns, or component orientation. Nevertheless, Stroustrup's unique relation to C++ and his many years of experience as a software developer provided him with more than enough valuable material and insights to share with the readers. In my opinion, junior developers will profit most from this part, but it is definitely of interest to senior developers as well.

CONCLUSION

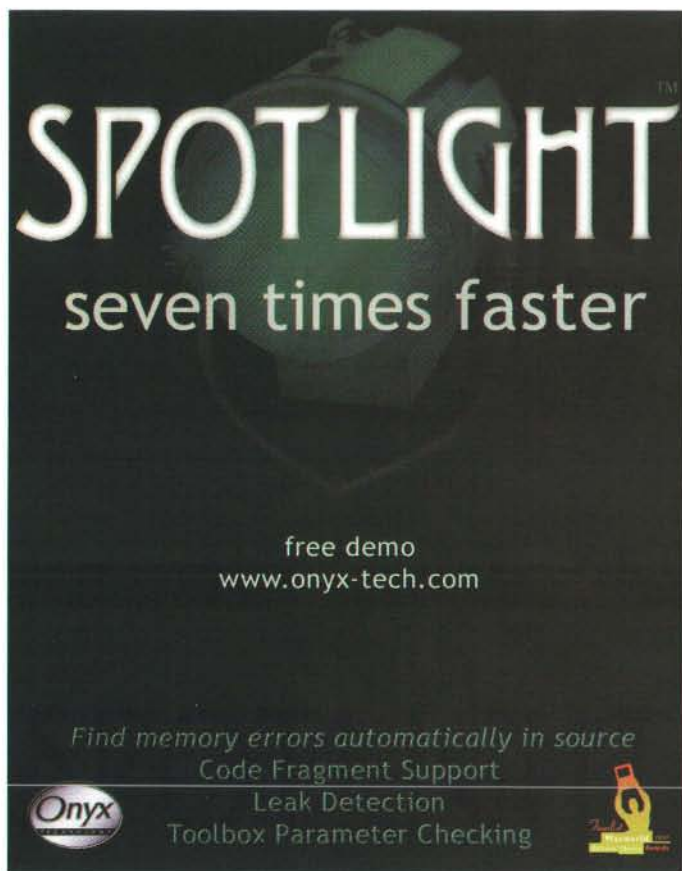
In the course of the last five years, I have read quite a few books in the field of computer science & engineering. Naturally, I encountered bad, good, and also some very good ones.

Bjarne Stroustrup's *The C++ Programming Language* is one of those very good books. It is crammed with useful information; to such an extent that — at least in the beginning — one learns more about programming in general and about C++ in particular every time one re-reads a section or a chapter.

To cut a long story short, *The C++ Programming Language* is not only a must-read, it is a must-have for every serious C++ programmer.

REFERENCES

- [1] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading (Massachusetts), 3rd edition 1997.
- [2] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading (Massachusetts), special edition 2000.
- [3] B. Stroustrup. *Bjarne Stroustrup. Home Page*. Located at <<http://www.research.att.com/~bs/>>.



SPOTLIGHT
seven times faster

free demo
www.onyx-tech.com

Find memory errors automatically in source
Code Fragment Support
Leak Detection
Toolbox Parameter Checking

Onyx

Research

Do all your holiday Shopping in ONE place!



This year...

***Get them something
they actually can USE!***



USEFUL GIFTS AND GADGETS! SM

PO Box 5200 Westlake Village, CA • 91359-5200 • info@radgad.com
Toll Free: 877/5-RADGAD • Outside U.S. Canada: 805/494-9797 • Fax: 805/494-9798



List of Advertisers

4D	IFC
Active Concepts	55
AD Software	26
Aladdin Knowledge Systems	5
Aladdin Systems	75
Belkin Components	71
Blue Dog	41
Blueworld	21
Bowers Development	47
Catalog Stuff	82
Concept Draw	19
Developer Depot	28-29
Digital Forest	13
Drive Savers	80
FairCom Corporation	7
Farallon	73
IDG	78-79
Intego	37
MacTech Central	83
Main Event	77
Mathemaesthetics	1
Metrowerks	BC
Movie Depot	81
Onyx Technologies	86
OpenBase International	45
Page Planet	33
Paradigma	23
PowerMax	9
RadGad.com	87
REAL Software	51
Register.com	61
Scientific Placement	38
SGI	11
Stone Tablet	35
SuSE	IBC
Technosoft	85
TerraSoft Solutions	74
Thinking Home	59
True Basic	57
UNI SOFTWARE PLUS	53
Vertical Solutions	31
VST Technologies	15
WebEx	17

Mac OS X Porting and Development Showcase

Art and Logic	66
Omni Group, The	69
Prosoft Engineering, Inc.	64
Recosoft	65
Red Rock Software	68
Robosoft	70
Shadetree	67

List of Products

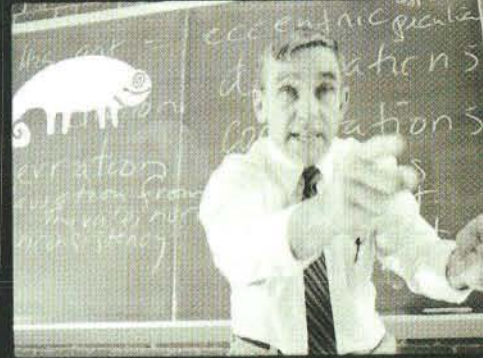
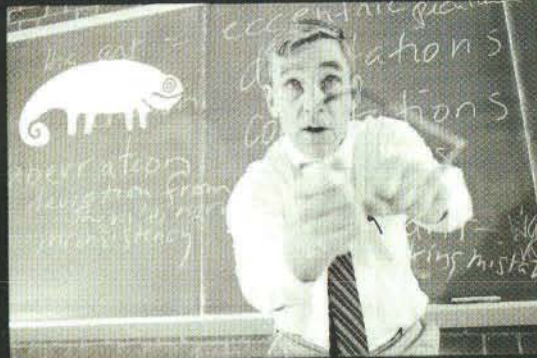
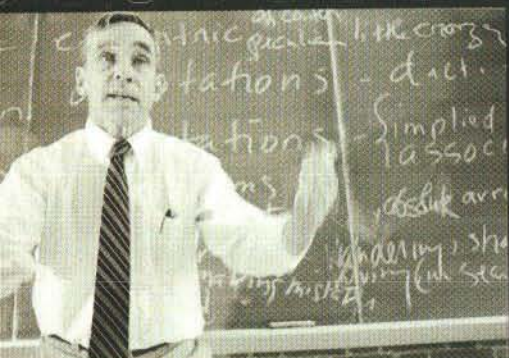
4D and WebSTAR • 4D	IFC
Application Service Provider • Blue Dog	41
AppMaker • Bowers Development	47
CodeWarrior • Metrowerks	BC
Contract Developer • Vertical Solutions	31
c-tree Plus • FairCom Corporation	7
Data Recovery • Drive Savers	80
Development Tools • Developer Depot	28-29
Domain Name Registration • Register.com	61
Equipment • PowerMax	9
FireWire Components • VST Technologies	15
Flat Panel Display • SGI	11
Funnel Web 3 • Active Concepts	55
Gifts, Gadgets and More • RadGad.com	87
Hardware • Belkin Components	71
Installer Maker • Aladdin Systems	75
Internet Service Provider • Digital Forest	13
Job Placement • Scientific Placement	38
Lasso Web Data Engine • Blueworld	21
Linux • SuSE	IBC
MacHasp USB • Aladdin Knowledge Systems	5
MGI • Page Planet	33
Multi-Media Communication • WebEx	17
NetBarrier • Intego	37
OO File • AD Software	26
Professional Macintosh and Internet Development • Thinking Home	59
RAD Studio • OpenBase International	45
REALbasic • REALSoftware	51
Resorcerer • Mathmaesthetics	1
Scripter • Main Event	77
Services-Internet • Movie Depot	81
SkyLINE 11MB • Farallon	73
Spotlight • Onyx Technologies	86
Stone Table • Stone Table Publishing	35
Trade Show/Conference • MacTech Central	83
Trade Show - Macworld SF 2001 • IDG	78-79
True Basic • TrueBasic	57
USB Stuff, FireWire Stuff • Catalog Stuff	80
Valentina • Paradigma	23
VOODOO Server • UNI SOFTWARE PLUS	53
Yellow Dog Linux • TerraSoft	74

Mac OS X Porting and Development Showcase

Programming Service • Prosoft Engineering, Inc.	64
Customised Computer Software Applications • Recosoft	65
Software Engineering Company • Art and Logic	66
Software Development Outsourcing • Shadetree	67
OS X Development • Red Rock Software	68
Consultants • The Omni Group	69
Porting and Implementation • Robosoft	70

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

HOW DO YOU SAY SuPERB IN LINuX?



SuSE.

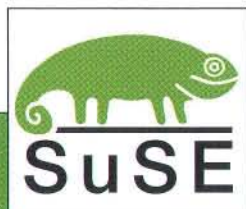
{soo' sah} is {soo' pûrb}

When you think of Superb, you think of unusually high quality. Like a superb wine, for example. Majestic. Rich. Luxurious. Superior. ■ SuSE Linux is all that. And more. More experience. More adaptability. More applications—over 1500. ■ More power to you. And more freedom, too. ■ No wonder more than 50,000 enterprises worldwide bank on its superb open source solutions. Making SuSE the international Linux leader—setting a higher standard for excellence, simplicity and support. ■ Even the price is superb. ■ So, how do you say superb in Linux? SuSE. It's a lesson well learned.

www.SuSE.com

Versions for Intel, Alpha, and PowerPC

The freedom to change.
The power to change the Linux world.



FREE
Online Course!
Intro to
PowerPlant
enroll at:
www.CodeWarriorU.com



Keeping Mac dreams alive since 1993.

Two great operating systems,
one CodeWarrior.

Mac developers depended on CodeWarrior to make the platform architecture shift from 68K to PowerPC processors. Now CodeWarrior does it again, speeding your transition to the next new operating system. CodeWarrior for Mac OS, Version 6.0 supports development for both OS X

and Classic Mac operating systems from a single, powerful, award-winning Integrated Development Environment. Discover how CodeWarrior for Mac OS, Version 6.0 can help you realize your Mac development dreams. Visit www.metrowerks.com/go/mac.

CodeWarrior®

